TRABALHO DE GRADUAÇÃO

AVALIAÇÃO DE ALGORITMOS DE EXPLORAÇÃO DE AMBIENTES POR ROBÔS MÓVEIS

Matheus Abrantes Cerqueira

Brasília, dezembro de 2018

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

AVALIAÇÃO DE ALGORITMOS DE EXPLORAÇÃO DE AMBIENTES POR ROBÔS MÓVEIS

Matheus Abrantes Cerqueira

Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Geovany Araújo Borges, ENE/UnB _____
Orientador

Prof. João Yoshiyuki Ishihara _____
Examinador

Enga. Letícia Helena Silva Porto _____
Examinadora

Brasília, dezembro de 2018

FICHA CATALOGRÁFICA

CERQUEIRA, MATHEUS ABRANTES
Avaliação de Algoritmos de Exploração de Ambientes por Robôs Móveis,

[Distrito Federal] 2018.

xiv, 49p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2018). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Robôs Móveis	2.Exploração
3. Cobertura	4.Kinect
5. Robótica	6.ROS
I. Mecatrônica/FT/UnB	II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

CERQUEIRA, MATHEUS ABRANTES, (2018). Avaliação de Algoritmos de Exploração de Ambientes por Robôs Móveis. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$15, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 63p.

CESSÃO DE DIREITOS

AUTOR: Matheus Abrantes Cerqueira

TÍTULO DO TRABALHO DE GRADUAÇÃO: Avaliação de Algoritmos de Exploração de Ambientes por Robôs Móveis.

GRAU: Engenheiro ANO: 2018

Matheus Abrantes Cerqueira

Campus Darcy Ribeiro, SG-11, Universidade de Brasília.

70919-970 Brasília – DF – Brasil.

Dedicatória

à Luana Abrantes Roque, com maior amor do mundo.

Matheus Abrantes Cerqueira

Agradecimentos

Gostaria de agradecer a todas as pessoas que foram importantes para o presente trabalho, ao prof. Geovany Borges pelo apoio, tempo e conhecimento concedidos, que foram fundamentais em minha jornada, motivando e inspirando a trabalhar melhor. Gostaria de agradecer à banca, prof. João Ishihara e Letícia Porto, que ao aceitaram fazer parte de evento tão singular do curso e do meu desenvolvimento.

Além disso, gostaria de agradecer a todas as pessoas do Laboratório de Robótica e Automação, que corroboram para o LARA ser ambiente de trabalho memorável e acolhedor que é. Em especial, agradeço ao Gabriel Araújo, que me auxiliou em toda jornada com o AMORA e a robótica móvel.

Os amigos de estudo, Gustavo Fernandes, Aldegundes Ceza e Luiz Vitor Reis, agradeço pelas horas de estudo e "papo furado"pela UnB. Agradeço a todas as pessoas que ajudam a UnB a ser uma universidade melhor para todos, seja por seus empregos ou por seu apreço por ela.

Finalmente agradeço a minha mãe, Luana e minha vó, Aparecida, que são meus maiores exemplos, e que desde pequeno me mostraram a beleza da vida e me ensinaram a apreciar meu aprendizado.

Matheus Abrantes Cerqueira

RESUMO

Atividades como exploração e cobertura auxiliam robôs a descobrir o ambiente, maximizando a região coberta tanto fisicamente quanto sensorialmente. O foco desse trabalho é a avaliação de diferentes técnicas exploratórias com robôs móveis sem construção de mapa, tais técnicas são baseadas em direcionamento probabilístico. Para isso é utilizado um modelo de robô móvel *Pioneer 3-DX*, presente no LARA (Laboratório de Automação e Robótica), dotado de sensor visual *Microsoft Kinect* e utilizando o ambiente de desenvolvimento aberto ROS (*Robot Operating System*)

Palavras Chave: Exploração, Robôs Móveis, ROS, Kinect, Cobertura, Pioneer 3-DX.

ABSTRACT

The exploration and coverage problem help mobile robots to know their environment and maximize the total area which is covered by the body of robot or by their sensors. This work presents the evaluation of different exploration techniques of mobile robots without any map construction or representation, and these techniques are based in stochastic goals. For this purpose, is used a Pioneer 3-DX robot model, from the Automation and Robotics Laboratory (LARA), and is equipped with a Microsoft Kinect visual sensor and works with Robot Operating System (ROS) framework.

Keywords: Exploration, Mobile Robot, ROS, Kinect, Coverage, Pioneer 3-DX.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS

Símbolos Latinos

X, Y, Z	Eixos ortonormais de sistema coordenadas referencial	
x	Localização no eixo X do referencial global	[m]
y	Localização no eixo Y do referencial global	[m]
z	Localização no eixo Z do referencial global	[m]
v	Velocidade linear	[m/s]
C	Espaço de configurações do robô	
N	Lugares ou nós de um grafo	
A	Arestas de um grafo	
$G(N, A)$	Grafo formado por N e A	
$\mathbf{q}$	Configuração representada por um vetor	
q	Configuração em espaço de dimensão não especificada	
H	Histórico de configurações do robô	
l	Eixo das rodas do robô	[m]
d_{obs}	Distância do obstáculo ao robô	[m]
a	Ângulo de Abertura	[°]
h_{kinect}	Altura do Kinect no eixo Z	[m]
D_p	Distância percorrida pelo robô	[m]
A_c	Área de cobertura	$[m^2]$

Símbolos Gregos

θ	Ângulo de orientação do robô de acordo com referencial global	[rad]
ϕ	Ângulo de orientação de destino em referencial global	[rad]
ρ	Distância do robô ao destino	[m]
σ	Variância	
$\boldsymbol{\Sigma}$	Matriz de covariância	
$\boldsymbol{\Sigma_H}$	Matriz de covariância de H valores	
$\boldsymbol{\nu}$	Processo estocástico vetorial	
γ_x	Processo estocástico unidimensional em x	[m]
γ_y	Processo estocástico unidimensional em x	[m]
γ_θ	Processo estocástico unidimensional na orientação	[rad]
γ_r	Processo estocástico unidimensional de distância	[m]

Grupos Adimensionais

i, k	Contador

Subscritos

$free$	espaço livre
obs	espaço ocupado por um obstáculo
a	aleatório
p	proximidade geométrica
e	esquerdo
d	direito
g	destino
min	valor mínimo
0	valor inicial

Sobrescritos

$\cdot$	Variação temporal
$-$	Valor médio
T	Transposto

Siglas

AGV	Veículo Guiado Automaticamente -*Automated Guided Vehicle*
AMORA	*Autonomous Mobile Robot Algorithms*
LARA	*Laboratório de Automação e Robótica*
ROS	*Robotic Operating System*
GNT	Árvore de lacunas de navegação - *Gap Navigation Tree*
SLAM -	Localização e Mapeamento Simultâneos - *simultaneous localization and mapping*
RDT	*Rapidly exploring dense trees*
ARW	Passeio Aleatório Adaptativo - *Adaptive Random Walk*
PCL	*Point Cloud Library*
SP	Sorteio de Pontos (método)
RW	Passeio Aleatório - *Random Walk*

Capítulo 1

Introdução

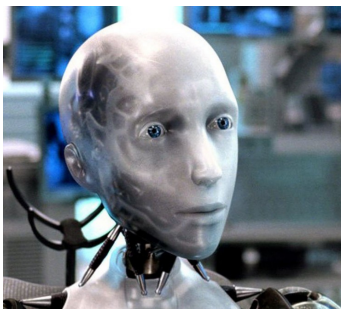
1.1 Contextualização

A robótica é um campo de reconhecimento do público geral, que em muitas vezes, alimentada por meios de comunicação e entretenimento, associam robôs a grandes projetos bioinspirados, com inteligência e em alguns casos até semelhantes ao homem. Tal associação, em muitos casos, provoca misticismo, admiração ou medo da robótica, fomentando discussões éticas e teorias sobre uma possível revolução de máquinas, vista em obras como *Eu robô* de Isaac Asimov ou *Exterminador do Futuro*(*The Terminator*) presentes nas Figuras 1.1(a) e 1.1(c).

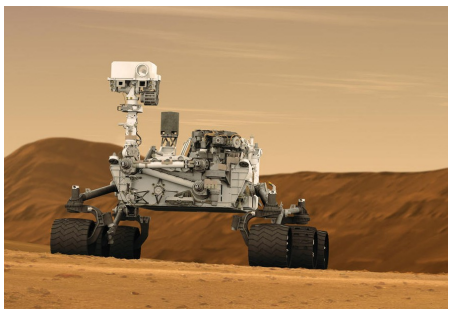
Porém, a evolução tecnológica fez com que a aproximação entre pessoas e robôs crescesse, os robôs já não ocupam somente as telas dos cinemas, mas também estão nas suas casas, escolas e até em outros planetas. Exemplos de tais robôs tem-se o Robô *Curiosity* de exploração planetária[1] e o *Nao Aldebaran*, robô utilizado tanto para pesquisa quanto para iteração homem-robô[2].

As aplicações da robótica são diversas, e em cada uma delas o robô opera inserido e de acordo com seu ambiente de trabalho, seja ele limitado por barreiras físicas externas como paredes ou internas como estar fixado em uma base, no caso de um manipulador. E em todos esses casos o robô deve ser capaz de explorar seu ambiente de trabalho, de forma a verificar por recursos, procurar por pessoas, ou somente aprender o ambiente.

[1] `https://www.nasa.gov/mission_pages/msl/overview/index.html`
[2] `http://dex.unb.br/noticias/172-extensao-em-pauta/490-unbeatables`



(a) Adaptação de Eu robô



(b) *Curiosity*

(c) Exterminador do Futuro

(d) Nao Aldebaran

Figura 1.1: Exemplos de robôs famosos

Da mesma forma operam os robôs móveis, ramo da robótica que possui diversas aplicações, tais como os destaques atuais de carros autônomos, AGVs (do inglês *Automated Guided Vehicle*) que operam em fábricas, até pequenos robôs didáticos e brinquedos. E de forma geral, os robôs móveis necessitam de ferramentas comuns tais como localização, mapeamento, detecção de obstáculos e planejamento de rotas.

Assim ao desenvolver uma atividade de exploração de forma acadêmica, espera-se contribuir com robôs móveis de diferentes complexidades, tais como limpeza, agricultura, segurança e resgate. Como exemplos dessas aplicações pode-se citar o robô aspirador, ROOMBA da *Irobot* e o robô de segurança K5 da *Knightscope* presentes na Figura 1.2, uma vez que tais robôs devem ser capazes de cobrir seu ambiente de trabalho como todo.

(a) Robô aspirador ROOMBA[3]

(b) Robô K5[4]

Figura 1.2: Roomba e K5

O AMORA (Autonomous Mobile Robot Algorithms), projeto de robótica móvel do LARA (Laboratório de Automação e Robótica) possui como foco o desenvolvimento de aplicações para robôs móveis, e assim, espera-se, a partir de um time de robôs móveis, colaborar com o desenvolvimento acadêmico da robótica móvel, colaborando com a evolução das técnicas pertencentes aos muitos robôs.

Nomeados de acordo com os três mosqueteiros, o AMORA conta com três robôs do tipo *Pioneer* da *Mobile Robots*, sendo um modelo 3-DX (Aramis), com direção diferencial em duas rodas e uma terceira de apoio, e dois modelos 3-AT (Porthos e Athos), modelos de quatro rodas e maiores que o 3-DX, porém com melhor capacidade para diversos terrenos.

Tais plataformas possuem grande capacidade para customização como exemplifica a Figura 1.3, com a presença de um manipulador, bem como de estruturas metálicas personalizadas com câmeras, caixas de som e monitores. Tal potencial para customização, juntamente com sua simplicidade e facilidade de uso fizeram da plataforma *pioneer* popular nas pesquisas em robótica móvel, nos mais diversos campos de atuação.

[4]https://www.irobot.com/for-the-home/vacuuming/roomba
[4]https://www.knightscope.com/knightscope-k5

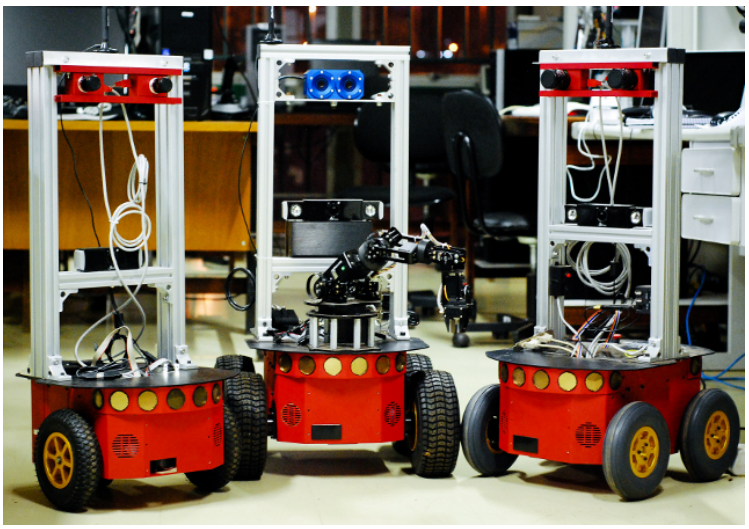


Figura 1.3: Grupo de robôs do AMORA, os três mosqueteiros[5]

1.1.1 Sensores

O sensor Kinect, com o nome original de "project natal", foi inicialmente desenvolvido para captar posturas e gestos humanos em jogos, sua proposta era possibilitar a jogadores interação e controle com videogames somente usando gestos e voz, sem a necessidade de utilizar controles.

Pouco tempo após seu lançamento foram criadas bibliotecas que dão,a programadores, habilidade de obter dados de distância fornecidos pela câmera infravermelho e imagem colorida por meio da câmera colorida. Além disso, seu baixo custo comparado com outras câmeras proporcionaram à comunidade científica novos horizontes, gerando um grande número de artigos em seus primeiros anos [4].

O Kinect possui um acelerômetro, 4 microfones, uma câmera e um conjunto foto emissor/receptor responsável pela informação de distância, o que fazem do Kinect um sensor útil no desenvolvimento acadêmico, seja para reabilitação/fisioterapia [5][6], ou para a robótica em si, utilizando informação de distância ou de sua câmera colorida [7].

Além do Kinect, o robô utilizado conta com 8 sonares frontais que foram utilizados para evitar a colisão de objetos que estejam mais próximos do chão. Os sensores estão afastados de $-90°$ a $90°$ intervalados de $20°$, como mostra a Figura 1.4. E cada sonar possui faixa de operação de 10 centímetros a 4 metros, aproximadamente, dependendo de sua taxa de amostragem.

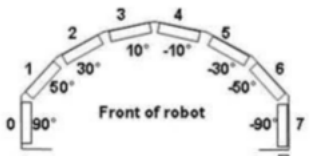


Figura 1.4: Disposição dos Sonares [1]

[5]https://amora.readthedocs.io/en/latest/robots.html

1.1.2 Framework ROS

A obtenção dos dados, simulação e desenvolvimento, foi feita utilizando o ambiente de trabalho para robôs ROS (*Robot Operating System*). ROS é um ambiente de trabalho para robôs de forma *open source*, ou seja, softwares que podem ser usados e compartilhados de forma livre, permitindo aprimoramento de técnicas pela comunidade de desenvolvedores e contribuindo para projetos maiores e mais robustos.[8]

O padrão básico de desenvolvimento do ROS utiliza de nós que se comunicam por tópicos ou serviços. No método de publicador/subscritor (*publisher/subscriber*), um nó pode publicar mensagens em um tópico e escutar mensagens de outros tópicos. A publicação em um tópico é feita por meio de padrões presentes em bibliotecas de mensagens, tais como *inteiro 8 bits*, *pose*, *nuvem de pontos*, dentre outras. Além disso, vários nós podem escutar o mesmo tópico, por exemplo uma mensagem de um sensor pode ser usada tanto pelo nó de localização quanto pelo nó de detecção de obstáculos.

Diferentemente da comunicação por tópicos, há a comunicação por meio de serviços, que implementam comunicações entre nós da forma cliente e produtor, em que um cliente faz uma requisição e obtém uma resposta de um produtor. A Figura 1.5 apresenta a topologia básica de comunicação por nós, em que um nó pode publicar um tópico para os demais nós, ou pode invocar um serviço diretamente a outro nó.

O desenvolvimento de acordo com a estrutura básica da Figura 1.5 corrobora para a modularidade e compartilhamento do ROS, uma vez que basta saber a descrição dos módulos e suas interfaces, para compartilhar e utilizar nós da comunidade de desenvolvedores e assim, a partir do compartilhamento de módulos básicos pode-se desenvolver aplicações cada vez mais elaboradas e complexas.

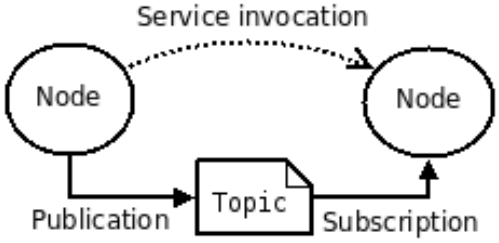


Figura 1.5: Esquema Básico do ROS

O ROS apresenta outras facilidades, como uma descrição do robô em arquivo de texto capaz de ser interpretada por máquina, descrevendo os modelo do robô tais tamanho, posição de sensores e outros componentes. Além disso, há ferramentas de visualização e integração com simuladores e outras bibliotecas tais como OpenCV,de visão computacional[6], e PCL(*Point Cloud Library*) de manipulação de nuvem de pontos[7]. A Figura 1.6 apresenta o robô real, seguido de sua estrutura obtida pelo código do modelo e a árvore de transformação, que na última figura apresenta a

[6]https://opencv.org/
[7]http://wiki.ros.org/pcl

hierarquia de conexão por meio de flechas, em que cada conexão contém uma transformação de coordenadas, permitindo rastrear qualquer referencial de forma simples, ou transformar informações de sensores para outras coordenadas.

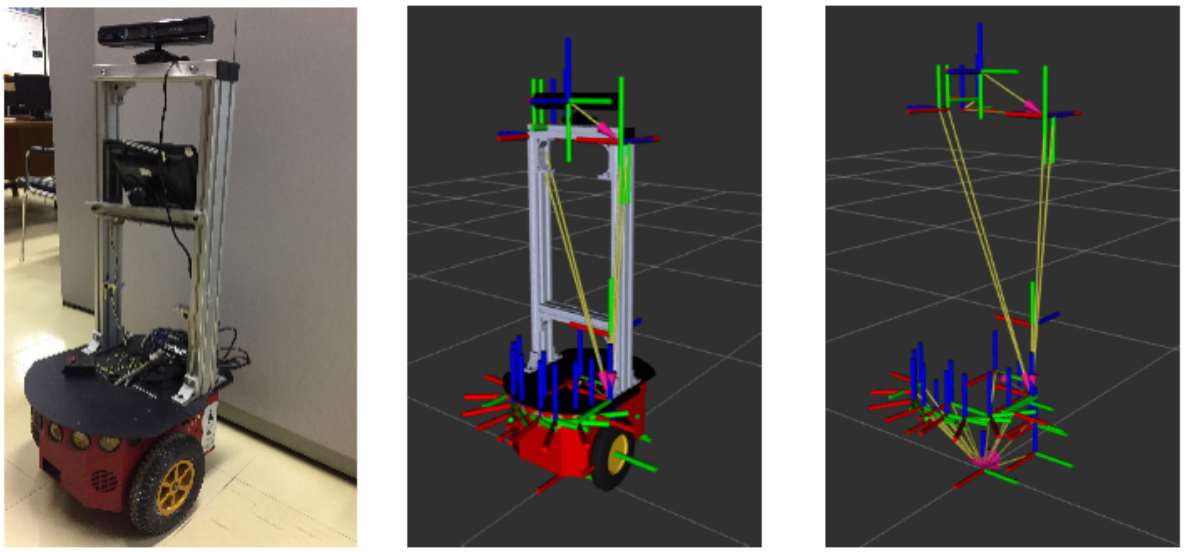


Figura 1.6: Aramis, visualização do modelo e árvore de transformação

1.2 Exploração

Atualmente muitos trabalhos focam em exploração ou cobertura, atividades que auxiliam em casos onde há necessidade de navegar de forma eficiente em ambientes desconhecidos, cobrindo a maior área possível. Um dos métodos tidos como clássicos é a exploração baseada em fronteiras (*Frontier Base Exploration*) feita inicialmente por Brian Yamauchi [9] e que já foi expandida para vários robôs [10]. Aqui o robô explora o ambiente ao se direcionar para fronteiras, que são as bordas entre regiões conhecidas e desconhecidas do mapa, e dessa forma o robô ganha informação sobre o ambiente.

Assim como Yamauchi, que associa explorar com a habilidade de construir um mapa, pode-se utilizar outras técnicas para direcionar o robô, como por exemplo o ganho de informação e entropia sensorial [11]. Além disso, até a própria exploração por fronteiras abre brechas para diferentes escolhas de fronteiras, tais como a escolha aleatória e a fronteira mais próxima, que apresentam diferenças no caminho percorrido [12].

1.2.1 Abordagens topológicas

Mesmo que Yamauchi e outros autores utilizem a exploração associada ao mapeamento métrico, outros autores não fazem, como é o caso de [13], os autores realizam a exploração guiada por um sensor visual, em que o robô, ao explorar, cria uma estrutura na forma de grafo que é utilizada para guia-lo.

Este grafo é criado a partir de obstáculos que tenham ao menos um marcador (*landmarks*), em que são utilizadas as bordas do obstáculo como nós do grafo, chamadas de lugar limite (*boundary place*). Já os marcadores consistem de objetos distintos no cenário, tais como cadeiras, mesas e latas de lixo. As ligações dos lugares limite são criadas toda vez que um marcador de um lugar limite B pode ser visto do lugar limite A e assim existe um espaço navegável que leva de A para B.

A exploração é realizada utilizando o grafo e movimentações simples, tais como circunavegar o obstáculo, verificando novos marcadores e em seguida se direcionar a novos marcadores. Finalmente os autores realizam testes com robô em escritórios e afirmam que foi possível explorar o ambiente em meia hora [13].

De fato a melhor contribuição do trabalho consiste na abordagem sem necessidade de localização ou de representações métricas. Porém o trabalho pressupõe que os obstáculos contém marcadores fixos, o que nem sempre é verdade em ambientes reais, onde pessoas caminham e modificam o ambiente. Além disso, o robô utilizado necessita de visão omnidirecional, o que pode limitar a aplicação.

Outro trabalho que não utiliza localização e que utilizam marcadores é o trabalho de cobertura de [14]. Nesse trabalho os marcadores são depositados pelo robô, cada marcador é ativo e recomenda uma direção preferencial a ser explorada (Norte,Sul,Leste e Oeste). Os marcadores formam um grafo para navegação do robô, em que os autores compararam com o passeio aleatório (*Random Walk*) e com a busca em profundidade (*Depth First Search*). Os resultados em simulação são positivos, porém não foram realizados testes reais e o robô possui a limitação de depositar marcadores.

Ainda no contexto de navegação por grafo, no trabalho [15], a navegação e exploração é feita em estrutura de grafos de lacunas (*gaps*). Não são utilizadas informações de coordenada ou representações métricas, somente a estrutura chamada de árvore de lacunas de navegação, GNT (do inglês *Gap Navigation Tree*).

Um sensor abstrato é utilizado para detectar as lacunas, ou descontinuidades na informação de profundidade. Ao utilizar sensores simples e em menor número, o sistema economiza em energia e possui menor complexidade de modelo e maior confiabilidade [15].

Os autores realizaram simulações e experimentos reais utilizando um robô Pioneer 2-DX e ambientes do tipo simples e multi conectados. Porém não avaliaram ambientes com pequenas lacunas, o que pode ser problemático, pois ambientes como escritórios possuem pequenas lacunas como pernas de cadeira e mesas.

A estrutura GNT pode ser utilizada em conjunto com outras regras exploratórias, como a movimentação seguindo paredes (*Wall-following*) [16] que corrobora para um padrão de movimentação simplista. O trabalho [16] é uma extensão de [17], porém não mais considerando o robô como um ponto e assim levando em conta passagens estreitas em relação a dimensão do robô.

Os mapas topológicos são aqueles representados por meio de um grafo, que por sua vez, descreve um arranjo de nós conectados entre si por uma série de arestas. Normalmente os nós correspondem

a regiões que possuem informação sensorial homogênea e os arcos refletem a conexão entre essas regiões [18].

A representação topológica codifica o ambiente de forma mais compacta que os mapas métricos, por exemplo, enquanto uma grade de ocupação é representada por uma malha 2D cujo valor de cada célula remete sua ocupação, um mapa topológico pode representar o mesmo ambiente em apenas alguns nós e arestas. Tome o exemplo da Figura 1.7, em que há uma malha com uma representação métrica ao lado de uma representação topológica utilizando os cômodos como nós e suas ligações como arestas, necessitando de apenas cinco nós.

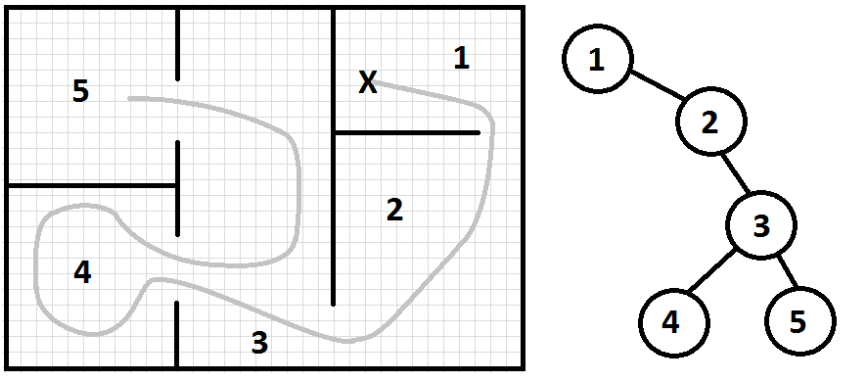


Figura 1.7: Exemplo de mapeamento topológico

Por mais simples que as representações topológicas possam ser se comparadas à representações métricas, tais representações também estão sujeitas a problemas tais como a dinâmica do ambiente, por exemplo se as características que configuram o ambiente topologicamente forem removidas ou ocorrer alguma oclusão a representação do mapa fica prejudicada. Além disso, o controle em mapa topológicos pode ser uma tarefa difícil se comparada com mapas métricos.

1.2.2 Padrões de movimentação

Estratégias reativas e bioinspiradas corroboram na solução de problemas, como por exemplo, o uso de memória espacial evita que robôs percorram áreas previamente visitadas e melhoram o direcionamento para sair de locais fechados [19].

Assim como uma trilha química de formigas, o robô manteria registros de quantas vezes passou por um determinado lugar e qual foi o tempo da última visita. Entretanto, manter tais registros podem implicar em malhas do ambiente, que com dito anteriormente, implicam em custo e podem divergir conforme a movimentação do robô.

A solução para o problema é manter registros locais, ao redor do robô, em que a odometria ainda é correta e também reduzindo o custo de armazenamento [19].

Uma possibilidade para cobertura é decompor a configuração espacial em células e percorrer cada célula com um padrão de movimento zig-zag (*Boustrophedon*) [20],[21]. Esse padrão é útil para cobertura em agricultura e limpeza, cuja motivação é movimentar-se por todos os pontos do

ambiente.

Em [22] a exploração é guiada por um padrão zig-zag e decomposição por nós e marcadores, entretanto diferentemente do trabalho [20], pois a cobertura consiste em percorrer o ambiente com os sensores, não necessariamente andar por todos os pontos, o que provocou uma melhoria na área coberta por distância percorrida.

Além disso, é utilizado mapa de tamanho fixo e centrado no robô, pois o robô é quadrúpede e necessita de informação sobre qualidade do caminho a ser percorrido, além disso, o mapa é utilizado para detectar marcadores que irão compor a abordagem topológica.

1.3 Objetivos

A partir dos diferentes usos e definições de exploração encontra-se em comum o problema de mover-se com a finalidade de conhecer o máximo possível sobre o ambiente, levando em conta o conhecimento atual sobre o ambiente e sem construção de representações globais. O modelo de robô utilizado foi o Aramis, robô de configuração diferencial, presente no LARA.

Para explorar, o robô deve ser capaz de tomar decisões a partir do conhecimento sobre sua vizinha, portanto, representações locais ajudam o robô a não colidir, além de ajuda-lo a evitar caminhos percorridos previamente. De forma a comparar diferentes algoritmos, utiliza-se a área explorada pelo robô, e devido a diferentes definições de exploração, utiliza-se tanto a área física coberta pelo robô além da área coberta pelos sensores do robô. A detecção de colisão é feita utilizando um sensor *Microsoft Kinect* e um conjunto de sonares.

1.4 Resultados

1.5 Apresentação do trabalho

Este relatório está organizado segundo cinco capítulos, em que no Capítulo 2 é apresentada a revisão literária sobre métodos de exploração utilizando direcionamento estocástico, bem como detecção de obstáculos. Em seguida, no Capítulo 3, está descrito o processo de desenvolvimento das técnicas exploratórias utilizadas nesse trabalho. No Capítulo 4 estão contidas os resultados de simulação e informações sobre o experimento com robô real. Finalmente, no Capítulo 5, estão presentes a conclusão do presente trabalho e suas perspectivas futuras. Em anexo, encontra-se a descrição do CD que contém a versão digital do trabalho.

Capítulo 2

Fundamentos

2.1 Introdução

Segundo Yamauchi, a exploração tem o potencial de libertar os robôs da limitação de navegar usando mapas previamente construídos. Nele o autor se refere a explorar como a capacidade que os robôs tem de navegar por ambientes desconhecidos e construir os seus próprios mapas [9]. Ao construir mapas o robô é capaz de detectar fronteiras, limites de espaço livre e áreas inexploradas, e mover-se até essas fronteiras. Pois dessa forma o robô estaria maximizando os ganhos de conhecimento sobre o mundo.

Observe que essa definição está intimamente ligada ao mapeamento, porém outros autores não associam explorar com construir representações métricas. Inclusive, assim como o foco desse trabalho, partem da situação de explorar sem mapas.

Algoritmos de mapeamento assumem que o robô pode criar uma representação métrica acurada do ambiente, o que pode ser problemático pois necessita de localização do robô em um referencial global, e assim desvios na posição serão refletidos na representação métrica criada.[13].

Uma opção para a dependência da localização é o Mapeamento e Localização Simultâneos ($SLAM$), tema ativo na comunidade científica, mas que não apresenta garantia de convergência em tempo hábil. Além disso, o $SLAM$ não resolve alguns problemas do mapeamento, tais como a grande quantidade de dados/complexidade computacional. Além de ser de difícil aplicação em ambientes externos e pouco estruturados [18].

Uma alternativa para a exploração que não utiliza nenhuma representação do ambiente global, seja ela métrica ou topológica, é utilizar de movimentações aleatórias.

2.2 Algoritmos de exploração de movimentação aleatória

Algoritmos de exploração podem ser englobados em categorias de planejamento de rotas, uma vez que seu objetivo é a aproximação de todos as configurações possíveis do robô [2]. Os algoritmos de planejamento de rota podem ser classificados dentre três categorias: algoritmos de mapa de

rotas, decomposição celular e funções potenciais [23]. Mapa de rotas armazenam a conectividade do espaço livre em um mapa topológico, criando caminhos para o robô, já a decomposição celular se utiliza da segmentação do espaço livre em estruturas que não se sobrepõe chamadas *células* que podem vir a ser conectadas em um grafo, já o método potencial utiliza de potencial para o direcionamento, onde o mesmo é repelido por obstáculos e atraído pelo seu destino.

Alguns algoritmos de exploração se fazem necessários de tais técnicas, como a utilização de mapas de rotas, Taylor e Kriegman utilizam de um grafo com landmarks [13], outros utilizam da GNT para a exploração e navegação [16], [13]. Além disso, Choset e Pignon[20] utilizam de um padrão de movimentação *zig-zag*, em conjunto com a decomposição celular para cobertura. Finalmente, campos potenciais são uteis também para a coordenação de exploração multi-robôs, utilizando um mestre com lei de controle diferente, reduzindo chances de mínimos locais presentes em abordagens por campos potenciais [24], além de ser possível usar campos para evitar que o robô passe por caminhos percorridos, igual a uma trilha química de formigas [19].

A Figura 2.1 exemplifica a utilização de campos potenciais e o seu uso na navegação, em que o destino atrai o robô e o um obstáculo que o repele. Tais campos potenciais resultam em uma 'força', resultando na trajetória presente na figura da esquerda.

O direcionamento do robô pode ser feito por meio de campos potenciais, em que o mundo é descrito por células e que cada célula exerce uma 'força' potencial ao robô, atraindo-o ou repelindo-o. De forma análoga ao robô, ao calcular seu potencial e descobrir sua força, guia-se o robô até um destino.

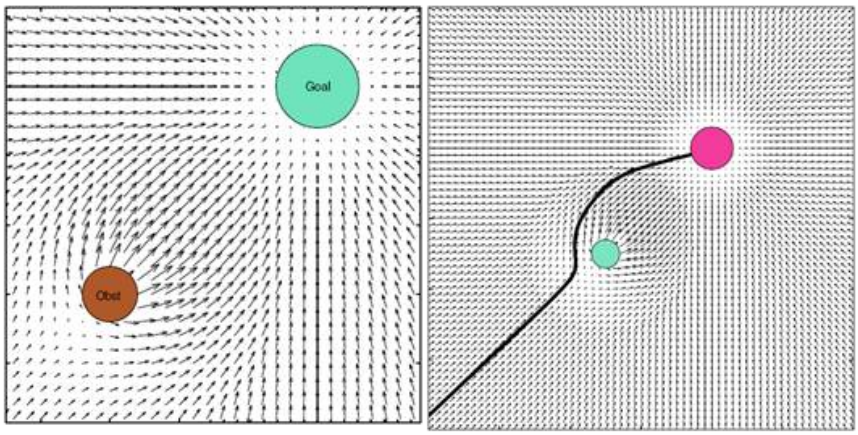


Figura 2.1: Campos potenciais para desvio de obstáculo e planejamento de rotas [1]

Além disso, os métodos de planejamento de rota podem ser definidos em outras categorias como o questionamento e entre métodos probabilísticos ou determinísticos. Métodos de questionamento único utilizam somente da requisição necessária, sendo mais simples usando menos tempo [3].

Existem muitas abordagens para fazer a exploração tais como fronteiras, utilizando grafos e outras, porém de forma simples pode-se fazer a exploração somente utilizando movimentação aleatória, seja por amostragem de pontos aleatórios, ou por campos potenciais aleatórios, que

[1] `Fonte:https://www.cs.mcgill.ca/~hsafad/robotics/#_msocom_4`

após certo período de tempo irá cobrir todo o ambiente [25]. Exemplos de exploração aleatória são movimentação por campos de atração aleatória ou por sorteio de ponto, como é o caso da RDT(*Rapidly exploring Dense Trees*) e RW (Random Walk), porém esses algoritmos são muito utilizados com auxílio de uma estrutura que grava os pontos, para planejamento de trajetória de questionamento único [2].

Seja o espaço total representado por C, formado pela união do espaço livre ao robô (C_{livre}) e o espaço ocupado (C_{obs}), a RDT é usada para indicar uma densa cobertura do espaço, em que a RDT é um grafo da forma $G(N, A)$, com os lugares N e as conexões A. Para tanto, ao tomar amostras aleatórias sobre o espaço livre C_{livre} e conetando a amostra $\boldsymbol{q}_a$ no ponto mais próximo da RDT a árvore cresce e cobre o espaço livre.

A Figura 2.2 apresenta resultados da exploração aleatória para uma configuração $C = [0, 1]^2$ em que a configuração inicial é $\boldsymbol{q}_0 = [0.5, 0.5]^T$, a imagem da esquerda possui 45 pontos enquanto a da direita possui 2345 pontos e é claro como o maior número de pontos leva a uma maior cobertura da área geral, porém um baixo número já pode ser suficiente para uma ideia geral do ambiente.

O algoritmo da RDT na inexistência de obstáculos depende de três passos básicos:

1. Gerar uma configuração aleatória $\boldsymbol{q}_a$ e adicioná-la como vértice;
2. Encontrar a configuração mais próxima $\boldsymbol{q}_p$;
3. Adicionar o vértice ligando ambas configurações ($\boldsymbol{q}_a$,$\boldsymbol{q}_p$).

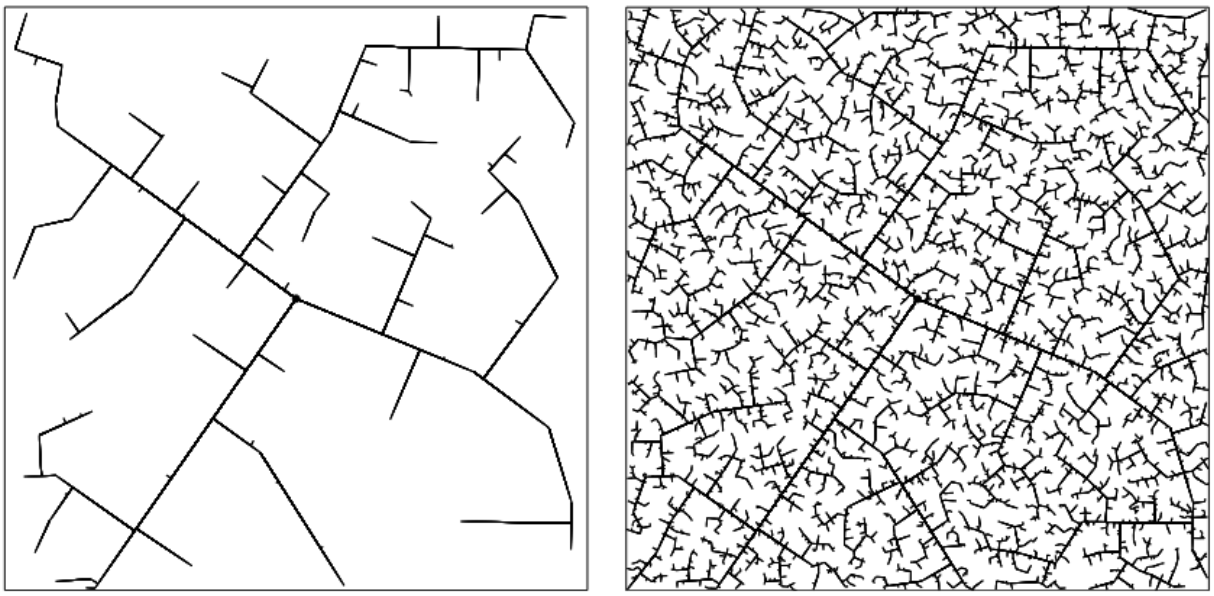


Figura 2.2: Exploração por RDT [2]

Já o algoritmo de passeio aleatório (*Random Walk*) utiliza de um processo estocástico de amostragem para evolução do processo. O termo *Random walk* foi primeira vez comentado em 1905, como processo matemático [26], o processo pode ser utilizado em diferentes abordagens tais

como modelagens ou de forma a percorrer grafos. O algoritmo pode ser usado para exploração ao tomar passos dentre uma probabilidade fixa, ou mudando sua probabilidade, com o Passeio Aleatório Adaptativo (ARW do inglês *Adaptive Random Walk*) [3].

Na exploração e cobertura o Passeio Aleatório é utilizado ao sortear ângulos que o robô deve se orientar e em seguida ao caminhar certa distância em linha reta. De forma geral, o robô sorteia pequenos desvios de ângulos até encontrar um obstáculo em que fará uma grande volta [27]. Morlok e Gini[28] utilizam de incrementos de [−10°,10°] a cada intervalo de tempo e ao encontrar um obstáculo o intervalo de sorteio passa para [120°,240°] e então volta para os pequenos incrementos.

Em vez de realizar passos com a mesma distribuição de probabilidade o ARW utiliza de um histórico para determinar a variância do próximo passo e dessa forma cria a capacidade adaptativa, em que durante espaços de configurações pequenos irá utilizar variâncias baixas e, evitando desperdiço de passos, já em espaços abertos sua variância será maior, produzindo passos mais largos. A Figura 2.3 apresenta tal adaptação da região 3σ de acordo com o passar do tempo.

O ARW possui eficiência em explorar o espaço de configurações e quando utilizado para planejamento de rota possui maior simplicidade se comparado com algoritmos como RDT, porém seu resultado produz caminhos de péssima qualidade, uma vez que gera ciclos desnecessários [3].

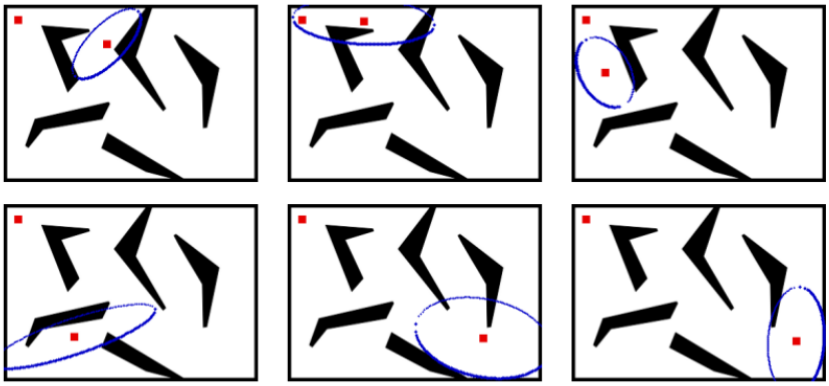


Figura 2.3: Algoritmo ARW com evolução da elipse 3σ [3]

Seja a configuração do robô no instante k, $\boldsymbol{q}_k$, a próxima configuração será dada por[29][3]:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \boldsymbol{\nu}_{k+1} \ , \tag{2.1}$$

em que $\boldsymbol{\nu}_k$ é da forma $\boldsymbol{\nu}_k \sim N(0, \boldsymbol{\Sigma}_k)$. Além disso, o processo da equação 2.1 se repete até o planejador de rota confirmar que $\boldsymbol{q}_{k+1}$ é um ponto válido. A matriz de covariância é descrita por

$$\boldsymbol{\Sigma}_{k+1} = max(\boldsymbol{\Sigma}_H, \boldsymbol{\Sigma}_{min}) \ , \tag{2.2}$$

em que $\boldsymbol{\Sigma}_{min}$, projeto do algoritmo, configura uma variância mínima e $\boldsymbol{\Sigma}_H$ é calculada utilizando os últimos H termos:

$$\mathbf{\Sigma}_H = \frac{1}{H}\left(\sum_{i=k-H}^{k-1} \boldsymbol{q}_i {\boldsymbol{q}_i}^T\right) + \bar{\boldsymbol{q}}_H \bar{\boldsymbol{q}}_H^T \tag{2.3}$$

O termo $\bar{\boldsymbol{q}}_H$ representa a média das últimas H configurações, ou

$$\bar{\boldsymbol{q}}_H = \frac{1}{H}\sum_{i=k-H}^{k-1} \bar{\boldsymbol{q}}_i \ . \tag{2.4}$$

Um detalhe importante é que a equação 2.2 pode ter sua forma alternada, utilizando as matrizes diagonais, ou

$$\mathbf{\Sigma}_{k+1} = max(diag(\mathbf{\Sigma}_H)), \mathbf{\Sigma}_{min}) \ , \tag{2.5}$$

e dessa forma, torna o método mais simples, porém com probabilidade não completa [29].

2.3 Detecção de obstáculos

Dada o destino por meio do algoritmo de exploração, o robô deve navegar de forma segura, tanto para si quanto para sua redondeza, evitando colidir com pessoas ou objetos, que podem ser prejudicados. Para isso o robô deve ser capaz de evitar colisões por meio de seus sensores.

O Kinect é um sensor de baixo custo capaz de fornecer dados relevantes para a navegação do robô, pois seja por imagens de distância, nuvem de pontos 3D, ou imagem *rgb,* o robô é capaz de ter informação rica do seu redor.

O Kinect é formado por um conjunto de sensores e atuadores, tais como uma câmera colorida, um conjunto emissor/receptor infravermelho, *array* de microfones, um motor de angulação e um acelerômetro, em que o conjunto emissor/receptor é responsável pela informação de distância fornecida pelo Kinect. Na Figura 2.4 é apresentado o arranjo de tais sensores e atuadores, mostrando a dispersão dos microfones e inclusive a posição do motor que controla a orientação do sensor geral.

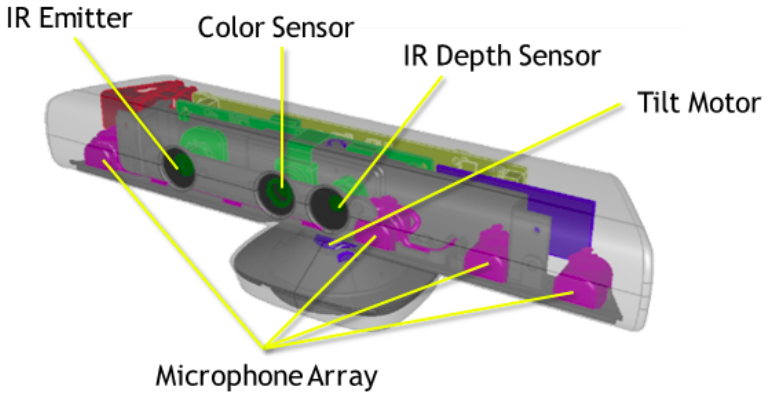


Figura 2.4: Esquema do Sensor Kinect [2]

As imagens do Kinect podem variar de resolução de acordo com a taxa de amostragem a tabela a seguir resume as características dos dados fornecidos, cuja imagens estão determinadas para taxa de 30 quadros por segundo(fps). Além disso, a faixa de operação do sensor de distância varia, por exemplo, quando conectado ao *video-game* sua faixa de operação é de cerca de 1.3m a 3.5m, já para algumas aplicações de desenvolvimento sua faixa é de 0.4-0.5m a 5-7m, essas e outras informações estão contidas na Tabela 2.1 [30] [31].

Dados	Características
Imagem colorida	$640x480$ pixeis a 30 fps
Imagem de distância	$640x480$ pixeis a 30 fps
Campo de visão horizontal	57°
Campo de visão vertical	43°
Faixa de operação de distância	cerca de 0.5 a 6m

Tabela 2.1: Informações do Kinect [30] [31]

Com o ROS, torna-se fácil adquirir imagem colorida, imagem de distância, nuvem de pontos e caso seja necessário, simplificar a nuvem de pontos/imagem de distância em um *laser scan*, que condensa a informação de distância no plano de um robô móvel terrestre. A Figura 2.5 apresenta dados obtidos com o Kinect e usando o ROS e a PCL.

A Figura 2.5 apresenta os dados de imagem colorida no canto superior esquerdo, a imagem de distância, no canto superior direito, e por fim, a nuvem de pontos. Ambos os dados foram obtidos em instantes de tempos próximos.

Na imagem de distância cada pixel possui um valor relativo a sua distância, tal imagem é representada na Figura 2.5, em que a imagem é apresentada de forma normalizada, ou seja, as maiores distâncias apresentam pixel com valor 1.0 (preto), enquanto as menores distâncias apresentam valor nulo (branco). Já a nuvem de pontos corresponde a uma série de pontos em que cada ponto possui um valor correspondente a sua configuração espacial (x,y,z).

[2]Fonte: Microsoft 2016

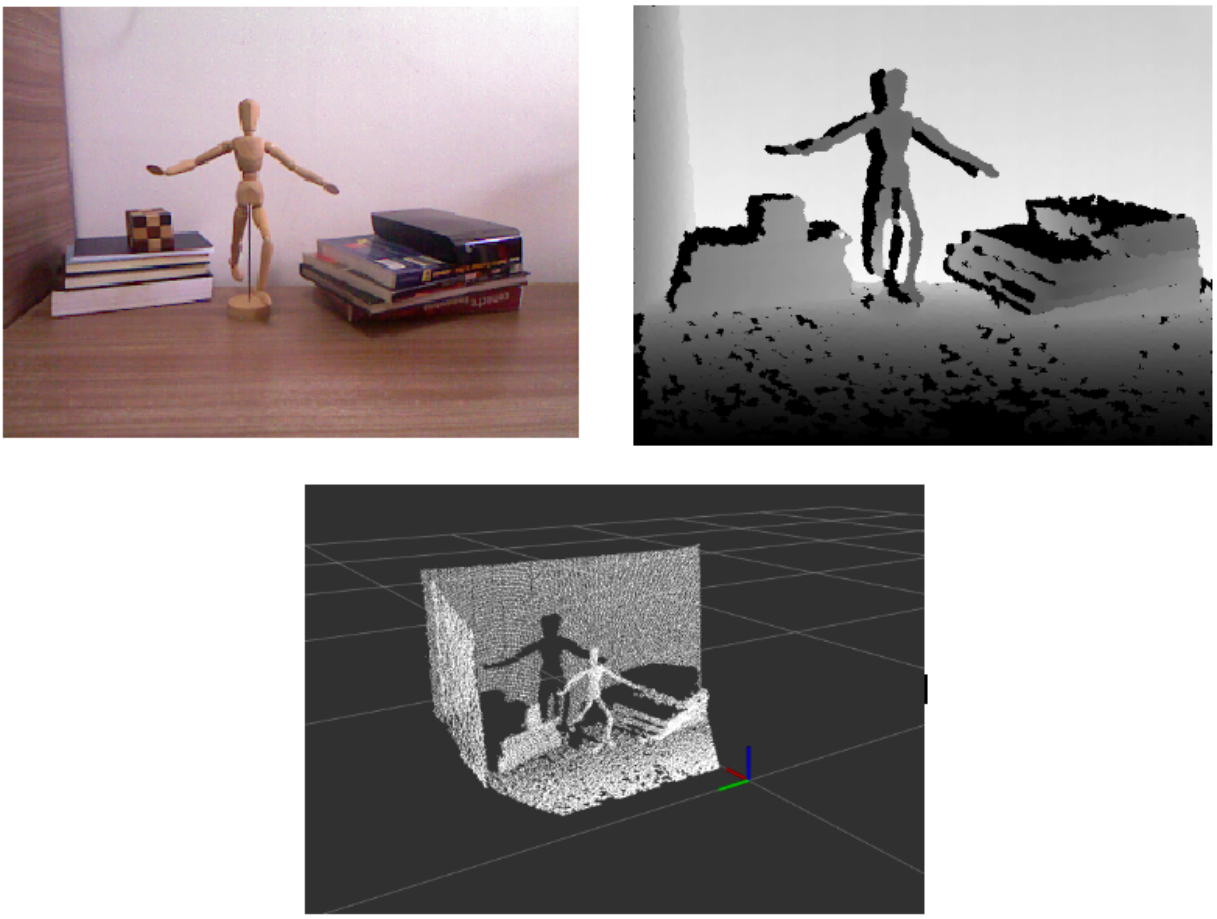


Figura 2.5: Dados do Kinect: imagem colorida, imagem de profundidade e nuvem de pontos

Dessa forma, o Kinect apresenta-se em um ótimo sensor de distâncias, porém uma vez ciente das suas limitações apresentadas na tabela 2.1 e tendo em mente sua posição no robô utilizado, Aramis(Figura 1.6), o Kinect possui uma distância mínima em que pode detectar um obstáculo mais próximo do chão, como é representado no desenho presente na Figura 2.6.

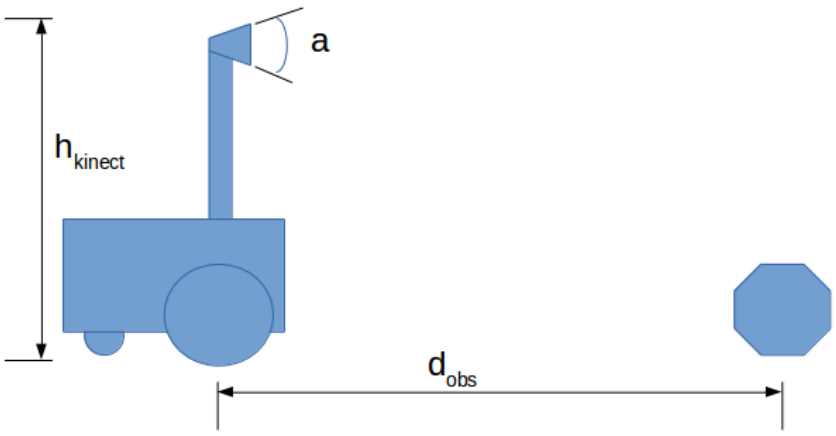


Figura 2.6: Esquema de detecção de um obstáculo

Tendo em vista que o campo de visão vertical é limitado em $a = 43°$, e que o Kinect está fixado a uma altura $h_{kinect} = 1m$, a distância mínima para o Kinect verificar um obstáculo próximo ao chão é de aproximadamente $d_{obs} = 2,54m$. E dessa forma, torna-se necessário a presença de um outro sensor de distância para evitar que obstáculos próximos ao corpo do robô não sejam detectados.

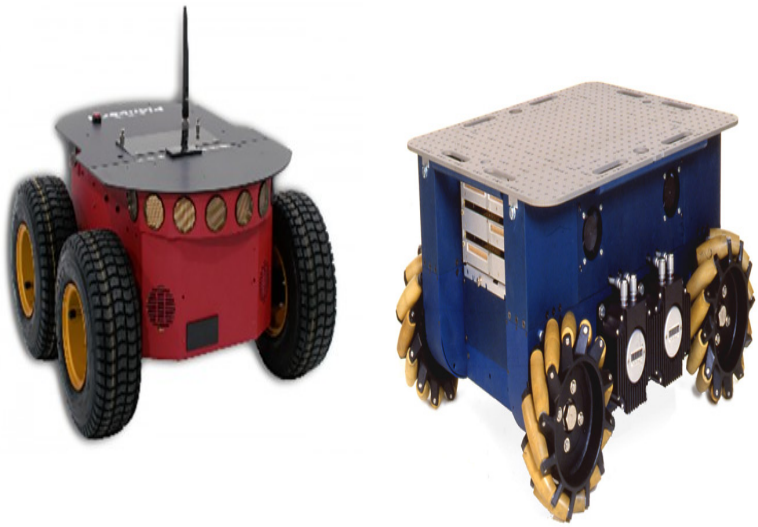
2.4 Configuração do Robô

Robôs móveis englobam uma gama robôs com as mais diferentes formas de locomoção, dentre os terrestres pode-se citar àqueles movidos por rodas e pernas, principalmente, porém outras configurações podem ser atingidas, tais como os robôs bioinspirados em cobras. Porém tais robôs possuem controle e modelagens mais complexas, levando a maior popularidade de robôs movidos por pernas e rodas, sobretudo movidos à roda.

Existe uma série de configurações possíveis para robôs móveis movidos por rodas, variando desde o número de rodas e seu arranjo, até a estrutura de tração. Começando pelo número de rodas, em termos de estabilidade estática, o número mínimo teórico de rodas é duas, porém o diâmetro de tais rodas deveria ser grande o que torna impraticável, portanto para uma configuração com duas rodas com tração requer um terceiro ponto de apoio, e assim, conforme o centro de massa esteja dentro do triangulo criado pelos três apoios a estabilidade estática é alcançada [32].

Mesmo fixado o número de rodas, existem diversas configurações tais como robôs com 4 rodas que não giram (*skid drive*), bem como robôs de 4 rodas mas com um par de rodas direcionável, assim como em carros (geometria de Ackermann), ou mais ainda, um robô de 4 rodas omnidirecional, que permite o robô mover-se no plano (x, y) em qualquer direção independentemente de sua orientação. Tais configurações possuem graus de facilidades em termos de controlabilidade e manobrabilidade, como por exemplo, robôs com geometria de Ackermann possuem manobrabilidade menor que àqueles com rodas que não giram (*skid*) [33].

E mais ainda, robôs omnidirecionais possuem maior manobrabilidade, porém a facilidade de manobrar possui, em geral, correlação inversa com a controlabilidade [32]. A Figura 2.7 possui um exemplo de robô omnidirecional, o Uranus da *Carnegie Mellon University* e como exemplo de um robô *Skid* tem-se o já citado Pioneer 3-AT.



(a) Pioneer 3-AT [3] (b) Robô Uranus [4]

Figura 2.7: Exemplos de robôs de 4 rodas com diferentes arquiteturas

Nesse contexto, algumas arquiteturas de robôs possuem melhores capacidades de manobra e são tidas como populares pela comunidade científica, mesmo que tal capacidade não se compare a de configurações omnidirecionais. Esse é o caso de configurações diferenciais passiveis de rotacionar em torno do seu próprio eixo [32], em que um movimento pode começar com uma rotação em seu próprio eixo para corrigir a orientação do robô. Além disso, tais configurações possuem boa capacidade de mover-se em linha reta, desde que a velocidade em cada roda seja a mesma.

Tais robôs de configuração diferencial possuem rodas com atuação desacoplada, podendo imprimir velocidades diferentes em cada roda. Tais modelos de velocidade seguem o esquema da Figura 2.8 e uma vez que suas rodas possuem o mesmo diâmetro e velocidades v_e e v_d para esquerda e direita, respectivamente, a velocidade linear do robô, v e sua velocidade angular $\dot{\theta}$,serão:

$$v = \frac{v_e + v_d}{2} \ , \tag{2.6}$$

$$\dot{\theta} = \frac{v_e - v_d}{l} \ , \tag{2.7}$$

$$\dot{x} = v \cdot cos(\theta) \ , \tag{2.8}$$

$$\dot{y} = v \cdot sin(\theta) \ . \tag{2.9}$$

[4]foto retirada de https://www.generationrobots.com/en/402397-robot-mobile-pioneer-3-at.html
[4]foto retirada de https://www.cs.cmu.edu/ gwp/robots/Uranus.htm

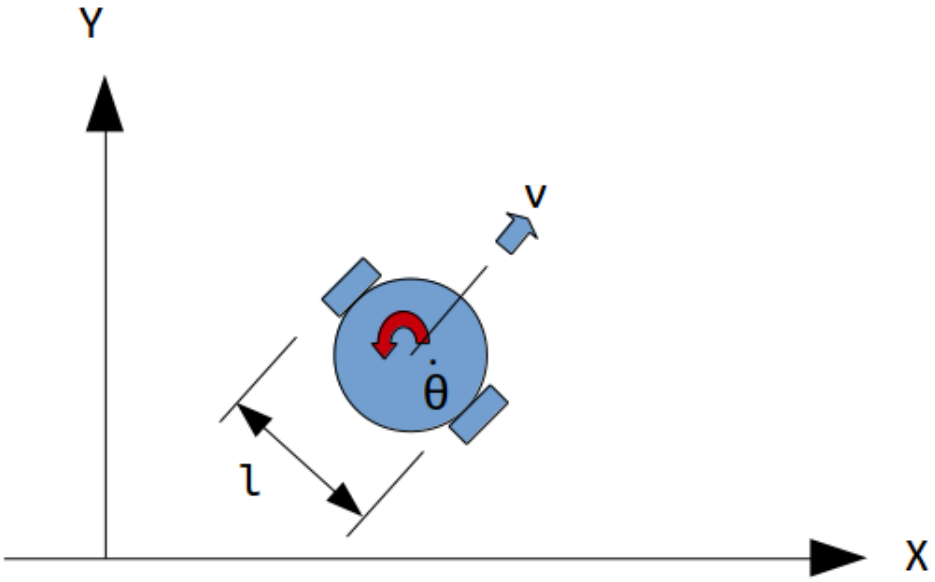


Figura 2.8: Modelo de velocidades do robô

A Figura 2.8 representa o robô e sua velocidades linear (v) e angular ($\dot{\theta}$), em que tais velocidades são geradas pelas velocidades de cada roda de acordo com as equações 2.6 e 2.7.

O ROS por sua vez, como ambiente de desenvolvimento de alto nível espera que o controle seja feito utilizando as velocidades v e $\dot{\theta}$, sua conversão para as velocidades de cada roda v_e e v_d é transparente ao programador e é realizada pelo microcontrolador interno do robô tomando o sistema das equações 2.6 e 2.7.

Capítulo 3

Desenvolvimento

3.1 Introdução

Os algoritmos de exploração seguiram o esquema da Figura 3.1, em que os módulos dos quadrados presentes em verde usam de dados providos pelas elipses em azul. Os destinos da exploração são gerados pelo módulo *gerenciador de destinos* a partir da configuração atual do robô e do seu histórico local. E uma vez definida a referência de destino, o módulo de controle é responsável por movimentar o robô, evitando colisões ao utilizar informação proveniente do módulo de *detecção de colisão*, este último que utiliza das informações do Kinect e do Sonar para verificar obstáculos.

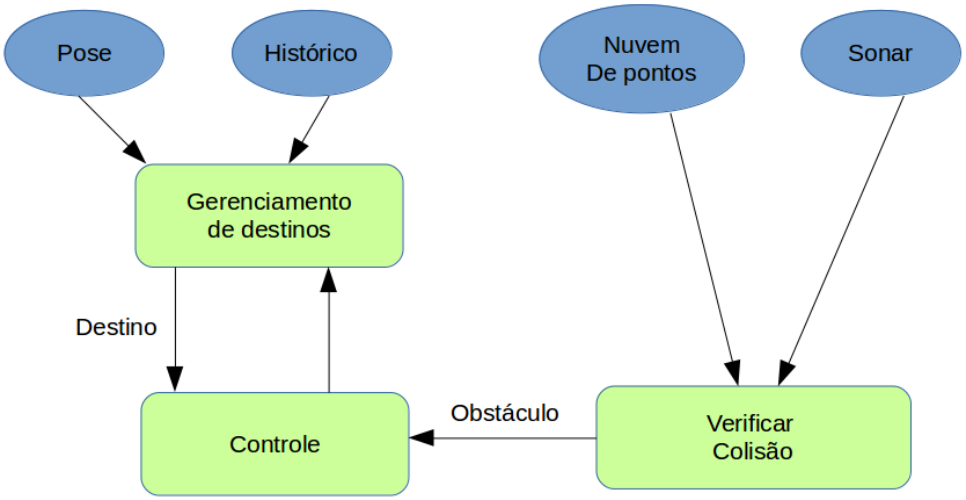


Figura 3.1: Estrutura dos algoritmos

Sobre o gerenciamento de destinos, uma vez que robôs podem usar de diferentes heurísticas exploratórias como seguir paredes e *zig-zag*, foram implementadas três tipos de estratégias baseadas em métodos aleatórios: *caminhada aleatória*, *sorteio de pontos* e *passeio aleatório adaptativo*.

A exploração por sorteio de pontos consistiu em sortear pontos aleatoriamente e fazer com que o robô vá até esse ponto, tais destinos são sorteados segundo distribuição uniforme em uma área

centrada no robô, idealmente seria desejável que a distribuição uniforme fosse fixa e englobasse todo o ambiente, porém devido a incapacidade de prever o tamanho do ambiente para definir a probabilidade e mais ainda, por desconhecer a presença de obstáculos, a área de distribuição uniforme de sorteio de pontos possui tamanho fixo e limitado.

Já para a caminhada aleatória, ocorre o sorteio de direções de exploração segundo distribuição uniforme. De maneira geral o robô sorteia pequenos desvios de direção como fator de inovação a cada distância percorrida até encontrar um obstáculo e aí o robô sorteará um desvio maior de exploração de forma e evitar o obstáculo.

Finalmente, o passeio aleatório adaptativo, sorteia destinos segundo distribuição normal, cuja adaptação reside na dependência da variância em função do histórico local de configurações do robô. Com isso espera-se que diferente dos métodos anteriores o ARW seja capaz de adaptar o sorteio de destinos conforme o espaço de configurações livres.

Os algoritmos seguiram a seguinte estrutura:

1. sorteie destino (x_g,y_g) não recente;
2. rotacione ao destino;
3. enquanto caminho estiver livre e não chegar ao destino:
   direcione-se ao destino.

3.2 Estrategias utilizadas

3.2.1 Sorteio de pontos

A Figura 3.2 apresenta o esquema de como é feito o sorteio de pontos para a exploração. O robô possui um histórico do seu passado local e uma janela cujo pontos serão sorteados, caso o ponto sorteado seja próximo ao histórico passado o robô irá sortear outro ponto. A figura exemplifica a região de sorteio pela área hachurada.

Na figura, o histórico está representado pelo retângulo branco dentro da área hachurada e foi implementado na forma de uma estrutura com 10 posições em que cada posição possui uma dupla (x, y) e cada posição possui distância de 10cm de suas vizinhas, dessa forma o último ponto está distante 1m do robô. Além disso, a verificação de novos destinos leva em conta uma região próxima aos pontos de "passado".

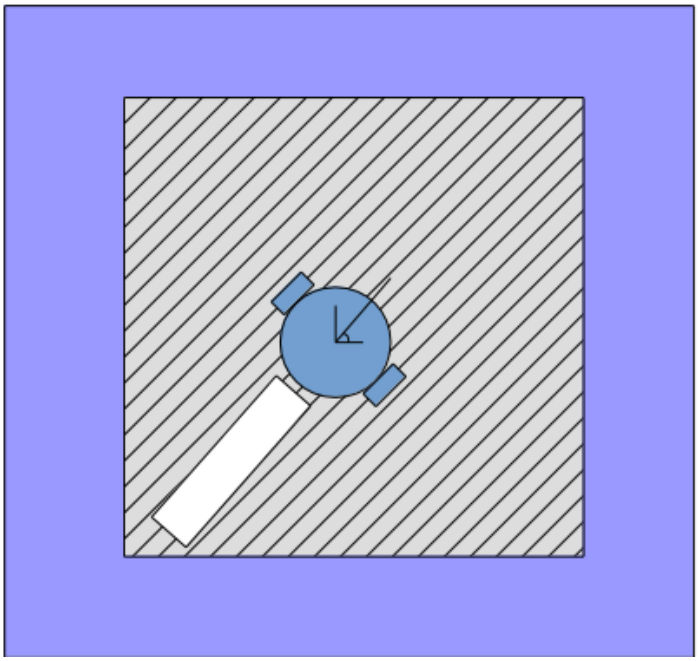


Figura 3.2: Esquema do sorteio de pontos uniforme

As informações do método SP estão representadas na Tabela 3.1, em que a distribuição de sorteio é uniforme e codifica a área hachurada da Figura 3.2, tal que $\gamma_x \sim U[-2.5, 2.5]$ e $\gamma_y \sim U[-2.5, 2.5]$.

Parâmetro	Valor
Número de pontos do passado	10
Distância dos pontos de passado	10cm
Distribuição do sorteio em x	$x + \gamma_x$
Distribuição do sorteio em y	$y + \gamma_y$

Tabela 3.1: Informações sorteio de pontos

O método de sorteio de pontos seguiu o algoritmo 1, que possui como entradas as definições do tamanho do histórico de passado e a posição inicial do robô que são armazenados na etapa inicial. Em seguida um laço faz o incremento do destino pelo processo estocástico γ até encontrar um destino válido. Uma definido um destino válido, um laço de repetição fará o controle do robô em duas etapas: *rotacione ao destino* e *caminhe até o destino*. Uma vez encontrado um obstáculo em seu caminho, ou seja, o caminho não está livre o robô volta a sortear um destino.

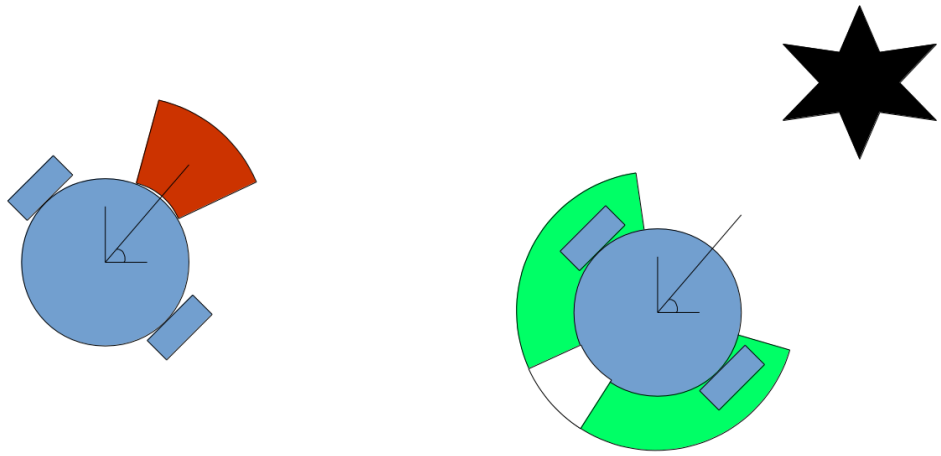
Algoritmo 1 Sorteio de Pontos

Entrada: $P, pose\_inicial$
Dados: $caminho\_livre, pose, destino, passado, \gamma$
início
  repita
    $tam(passado) \leftarrow P$
    $pose \leftarrow pose\_inicial$
    repita
      $destino \leftarrow pose + \gamma$
      $compara(destino, passado)$
    até *destino válido*;
    enquanto *caminho_livre or* !(*pose diferentedestino*) **faça**
      rotacione ao destino
      caminhe até o destino
    fim
  até ;
fim

3.2.2 Passeio Aleatório

A Figura 3.3 apresenta a forma de sorteio da exploração por RW, em que em uma condição livre de obstáculos o robô sorteia um ângulo de acordo com região vermelha da Figura 3.3(a) e também sorteia uma distância que deve explorar na direção já definida. Caso não encontre obstáculos o robô volta a sortear ângulos da mesma maneira. Porém, caso encontre um obstáculo, o robô ira entrar no modo de evitar obstáculo, como mostra a Figura 3.3(b), que possui um obstáculo representado pela estrela. Aqui o ângulo sorteado terá uma região maior, representada em verde, a menos da região em branca.

A recusa em ângulos da região branca menor da Figura 3.3(b) é para evitar que o robô volte pelo caminho que percorreu e assim encontrando novos caminhos.



(a) Sorteio do ângulo em condição livre

(b) Sorteio de ângulo na presença de um obstáculo

Figura 3.3: Esquema da caminhada aleatória

Os parâmetros que configuram o método do passeio aleatório estão contidos na Tabela 3.2, em que $\gamma_{\theta,free}$ representa a informação de sorteio em configuração livre de obstáculos e $\gamma_{\theta,free} \sim U[-\pi/10, \pi/10]$. Além disso, $\gamma_{\theta,obs}$ representa o sorteio no modo "evitar obstáculo"e $\gamma_{\theta,obs} \sim U[\pi/3, 5\pi/3]$. A região para evitar o passado é codificada por $180 \pm 18°$, ou $\pi \pm (\pi/10)$.

Parâmetro	Valor
Distribuição ao caminhar para frente	$\theta + \gamma_{\theta,free}$
Distribuição ao encontrar um obstáculo	$\theta + \gamma_{\theta,obs}$
Ângulos para evitar passado	$[\theta + 9\pi/10, \theta + 11\pi/10]$
Distribuição de distância percorrida	$\gamma_r U[0, 4]$m

Tabela 3.2: Parâmetros passeio aleatório

O algoritmo 2 apresenta o pseudocódigo do RW, que é próximo àquele utilizado no SP, porém com leves diferenças, tais como os processos de sorteio para condição livre, ou presença de obstáculo, bem como os destinos serem definidos por um ângulo e uma orientação ϕ, ρ.

Algoritmo 2 PASSEIO ALEATÓRIO

Entrada: $\nu_{obs}, \nu_{livre}, pose\_inicial$
Dados: $caminho\_livre, pose, \rho, \phi, passado$
início
  repita
    $pose \leftarrow pose\_inicial$
    se *caminho_livre* **então**
      $[\phi, \rho] \leftarrow pose + \nu_{livre}$
    senão
      $[\phi, \rho] \leftarrow pose + \nu_{obs}$
    fim
    enquanto *caminho_livre or* !(*pose diferente destino*) **faça**
      rotacione ao destino
      caminhe até o destino
    fim
  até *condição_parada*;
fim

3.2.3 Passeio Aleatório Adaptativo

Os parâmetros que configuram o passeio aleatório adaptativo estão descritos na tabela 3.3, destaca-se uma diferença dos métodos anteriores que é o fato do ARW possuir dois registros de pontos passados, porém com diferenças contextuais. Assim como os métodos anteriores, o ARW possui um registro de passado imediato, com 10 pontos, armazenando o último metro percorrido pelo robô. Diferentemente, o registro de tamanho H salva os últimos pontos sorteados pelo algoritmo do ARW, pontos esses que serão utilizados no cálculo da variância.

Da mesma forma que os anteriores, o equema de sorteio de pontos do ARW pode ser visto na Figura 3.4, em que a área hachurada representa o área de sorteio dos pontos e a região em branco o passado no qual são rejeitados pontos.

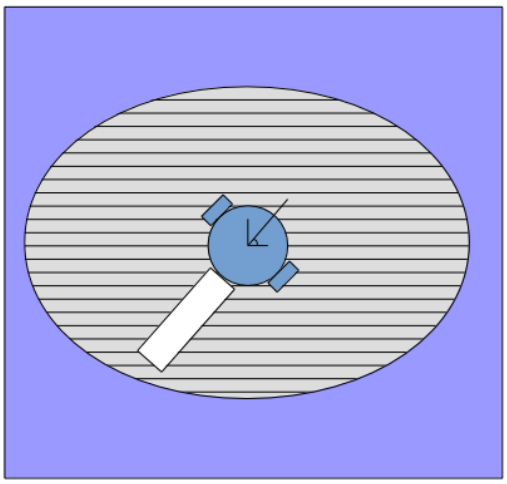


Figura 3.4: Esquema de destinos do ARW

Parâmetro	Valor
H	20
$\sigma_{x,min}$	1.0
$\sigma_{y,min}$	1.0
Tamanho passado	10
Espaçamento passado	10cm

Tabela 3.3: Parâmetros ARW

Novamente, o algoritmo do ARW é próximo dos algoritmos do RW e do SP, porém com as singularidades do ARW. O algoritmo 3 apresenta o algoritmo do ARW.

Algoritmo 3 Passeio aleatório adaptativo

Entrada: $\Sigma_{min}, pose\_inicial$
Dados: $caminho\_livre, pose, \rho, \phi, passado$
início
  repita
    $tam(passado) \leftarrow P$
    $pose \leftarrow pose\_inicial$
    repita
      $destino \leftarrow pose + \nu$
      $compara(destino, passado)$
    até *destino válido*;
    enquanto *caminho_livre or* $!(pose! = destino)$ **faça**
      rotacione ao destino
      caminhe até o destino
    fim
  até *condição_parada*;
fim

3.3 Movimentação e detecção de Obstáculos

Uma vez definido os destinos de exploração, o módulo de controle deve ser capaz de movimentar o robô até o mesmo alcançar seu objetivo, a menos que exista algum obstáculo em seu caminho. Para a movimentação, o robô é modelado como uma partícula, que possui configuração $\boldsymbol{q}$ da forma

$$\boldsymbol{q} = [x, y, \theta]^T \,, \tag{3.1}$$

dado um sistema de referências global, como mostra a Figura 3.5, um destino é representado pela estrela que possui coordenadas (x_g, y_g), o controle consistiu em rotacionar o robô até alinhar com o destino e em seguida faze-lo direcionar ao destino.

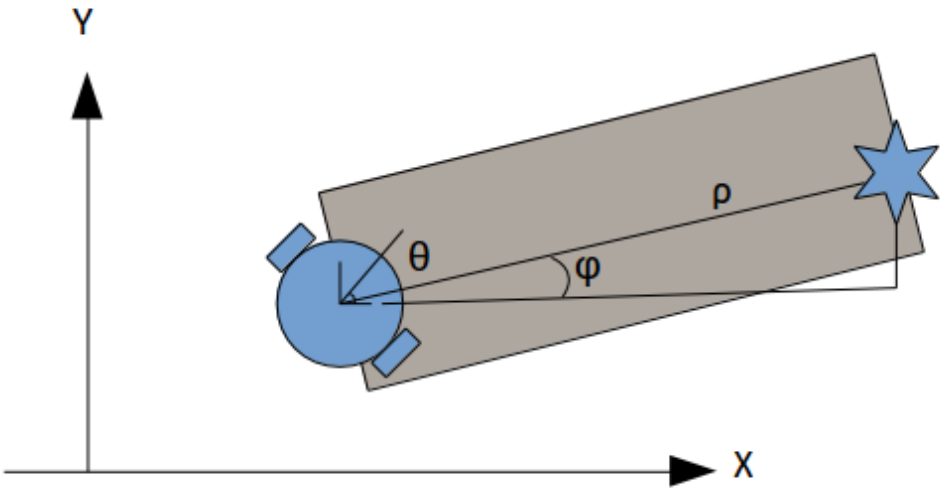


Figura 3.5: Configuração do robô e de um destino

Utilizando o ROS, a movimentação do robô é feita ao determinar sua velocidade linear v e sua velocidade rotacional $\dot{\theta}$. O robô imprimi velocidade rotacional até alinhar sua orientação com a orientação do obstáculo, ou seja $\theta = \phi$, em seguida caminha pela distância ρ, presente na Figura 3.5, e em ambos os casos foi utilizado um controle proporcional.

Como o robô é modelado por um ponto que é descrito pela configuração da equação 3.1, a verificação de colisão infla os obstáculos de forma a compensar o tamanho do robô, esse tratamento de inflar os obstáculos é observado pela região cinza da Figura 3.5, que é a região que o robô irá verificar por obstáculos, ou seja, a região que ele irá percorrer.

Para tal região foi utilizado o valor de inflação $0,7m$ ao redor de qualquer ponto de obstáculo, ou seja a região em cinza possui largura de $0,7m$, tal foi escolhido por englobar o maior diâmetro do robô, que é de $0,52m$, e também engloba a distância mínima do sensor Kinect como pode ser visto na Figura 2.1.

3.4 Ambientes de Simulação

A simulação foi conduzida utilizando o modelo do Aramis, juntamente com o ROS e o simulador Gazebo, simulador de código aberto que possui integração com o ROS, sendo seu simulador padrão. Por mais que sejam projetos separados, a integração é tal que permite o uso do gazebo em um mesmo arquivo *xml* de configurações.

Para verificar as características dos diferentes algoritmos utilizou-se quatro ambientes com pequenos acréscimos e diferenças entres eles, tais ambientes podem ser vistos na Figura 3.6.

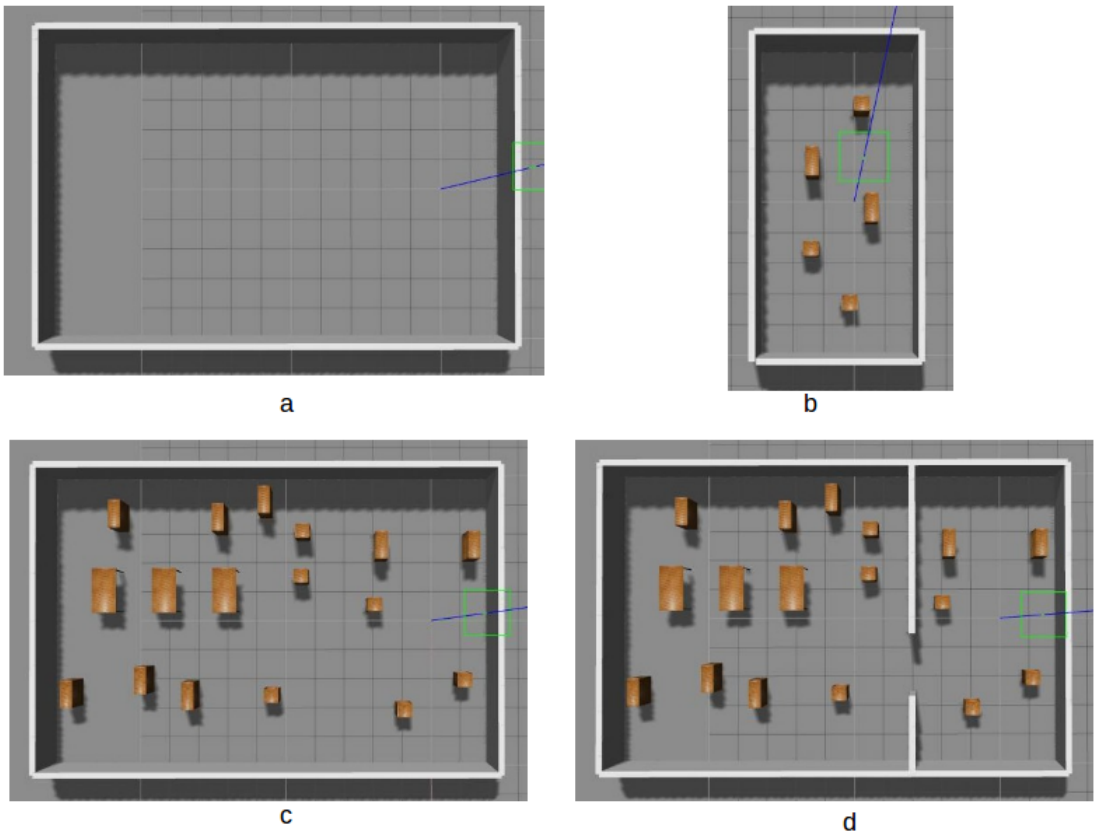


Figura 3.6: Ambientes de simulação

O primeiro ambiente utilizado consistiu em uma área vazia, porém limitada por quatro paredes e com isso espera-se verificar a capacidade geral dos algoritmos de explorar e uma vez que não há obstáculos, verificar a velocidade de exploração. Em seguida foi utilizado o ambiente da Figura 3.6.b, que diferente do primeiro ambiente, possui uma série de obstáculos, aqui espera-se observar a capacidade dos algoritmos de lidar com ambientes estruturados.

O terceiro ambiente consistiu na primeira configuração, porém com obstáculos para utilizar um ambiente mais real, tal como um escritório. Finalmente, na Figura 3.6.d utilizou-se o mesmo ambiente da etapa anterior, porém dividido em duas câmaras e aqui espera-se verificar a capacidade de explorar ambientes estruturados e com divisões.

Ambiente	Área total	Obstáculos	Câmaras
1	$15x10m^2$	Não	1
2	$5x10m^2$	Sim	1
3	$15x10m^2$	Sim	1
4	$15x10m^2$	Sim	2

Tabela 3.4: Resumo dos ambientes

3.5 Método de avaliação

O método de avaliação utilizado foi de comparar a área coberta pelo robô, tal área representa a área que o robô efetivamente visitou no ambiente. Para efeito de comparação foi utilizada a razão de tal área pela distância percorrida pelo robô, dessa forma o evento de parada da exploração foi o robô percorrer uma distância máxima definida previamente.

As distâncias máximas utilizadas foram de $50m$ e $100m$ e para o cálculo da área coberta, foi utilizado o maior raio que engloba o robô que vale $0,26cm$. A Figura 3.7 apresenta um esquema de como tal área e distância é percorrida, primeiro o robô percorre uma distância D_p e em ambiente externo, uma imagem que representa o mapa do ambiente possui seus pixeis alterados de acordo com a movimentação do robô, a área é então calculada ao contar os pixeis alterados e o fator de comparação é computado pela área de cobertura pela distância percorrida, ou A_c/D_p.

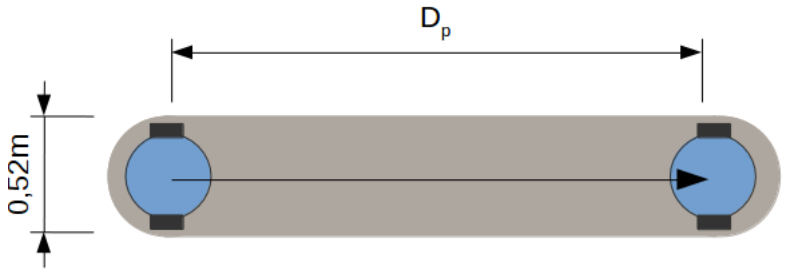


Figura 3.7: Método de avaliação

A Figura 3.7 apresenta um esquema da forma de avaliação, com a distância percorrida e a área coberta, A_c, representada pela área em cinza. Além disso, a Figura 3.8 apresenta exemplo de tal imagem para valores reais de simulação, a imagem é binária e os pixeis brancos correspondem à área coberta pelo robô, tal imagem foi feita posterior com os dados de simulação.

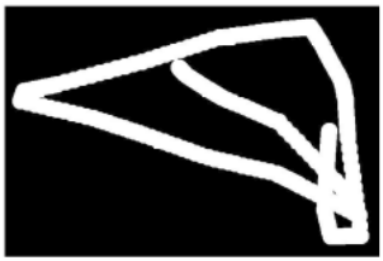


Figura 3.8: Exemplo de área coberta

3.6 Experimento real

O objetivo do experimento real foi verificar a exploração do robô em um ambiente de maior complexidade, porém nenhuma comparação entre as estratégias exploratórias foi realizada, uma vez que o robô não dispõe de localização para tal feito.

Capítulo 4

Resultados

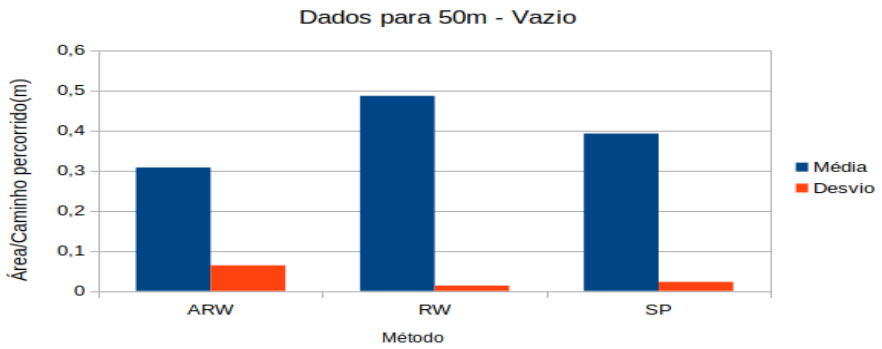
4.1 Simulação

Para cada combinação de ambiente e distância percorrida foram feitas 5 simulações e em seguida foi computado a área coberta pelo robô durante cada simulação. Finalmente, para a avaliação foram tiradas as médias e desvio padrão das simulações, mesmo com a distância fixa, ocorreu leves diferenças entre as simulações, portanto utilizou-se o valor da área coberta pela distância, o que ajuda a comparar mesmo algoritmo para diferentes distâncias.

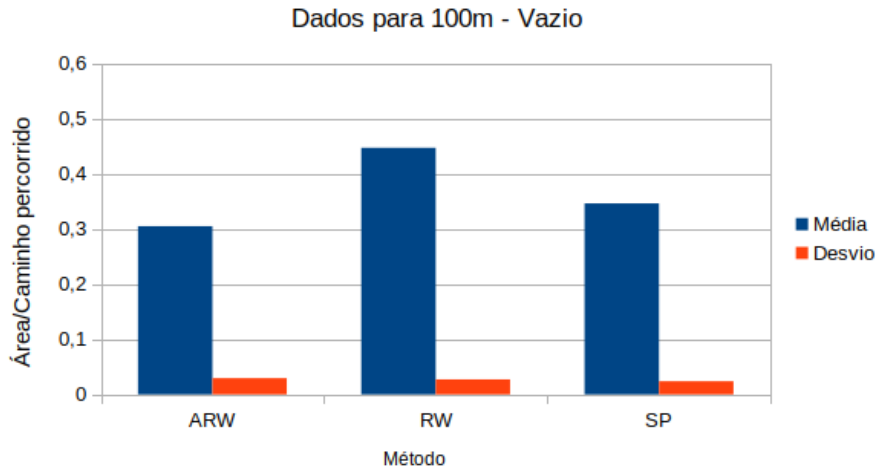
Para computar a área explorada foi utilizada uma imagem binária em que cada pixel possui resolução de 2cm e dessa forma, preenche-se cada pixel ocupado pelo robô, ao final o resultado será uma imagem binária em que cada pixel com valor 1 foi ocupado pelo robô e computa-se a área total explorada pelo robô. Para preencher a imagem, foi considerado o robô como um círculo cujo raio vale a metade da maior medida lateral do robô. Para as informações de pose do robô foi utilizado um *plugin* do Gazebo para gerar a informação de *Ground Truth*.

4.1.1 Ambiente livre

O primeiro ambiente, presente na Figura 3.6 não apresenta obstáculos, porém é limitado por paredes e espera-se ajudar a avaliar os algoritmos em condições sem obstáculos. A seguir estão presentes as tabelas com valores médios e desvio padrão, além de sua representação visual.



(a) Dados para distância fixa de 50m



(b) Dados para distância fixa de 100m

Figura 4.1: Dados para ambiente vazio

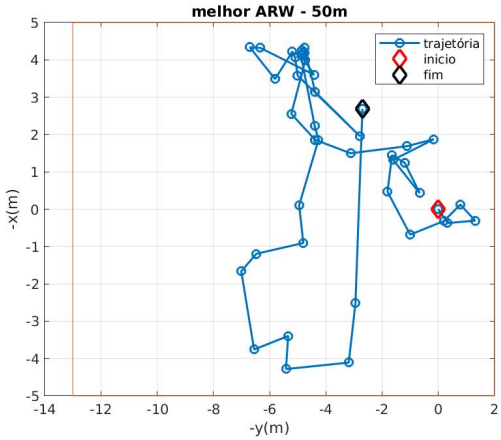
Método	Área/Distância para 50m	Área/Distância para 100m
ARW	$0.3076 \pm 0.0637m$	$0.3048 \pm 0.0294m$
RW	$0.4860 \pm 0.0134m$	$0.4468 \pm 0.0273m$
SP	$0.3921 \pm 0.0226m$	$0.3461 \pm 0.0238m$

Tabela 4.1: Valores médios para exploração do ambiente vazio

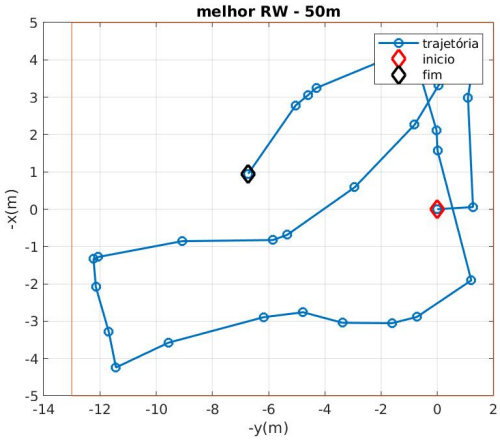
Os dados da Figura 4.1(a) mostram que de forma geral o algoritmo do RW obteve melhores resultados que os demais, uma vez que sua média foi mais elevada tanto para 50m quanto para 100m. Isso por que o RW, ao sortear pequenos ângulos incrementais na orientação do robô evita que o mesmo realiza muitos ciclos em sua trajetória. Isso pode ser visualizado ao comparar os melhores resultados dos algoritmos na Figura 4.2, observe como os algoritmos ARW e SP possuem destinos muito mais próximos e com ciclos e repassagens.

Além disso, ao comparar os valores presentes na tabela 4.1 é claro como os valores médios para os algoritmos RW e sorteio de pontos caíram ao aumentar o tamanho do percusso, diferente do

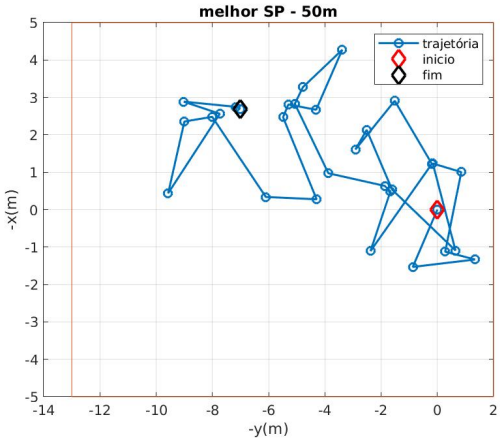
algoritmo ARW e isso se deve ao fato que o algoritmo possui um tempo necessário para aumentar adaptar seu número de adaptação devido ao histórico de tamanho H.



(a) Melhor trajetória para ARW



(b) Melhor trajetória para RW

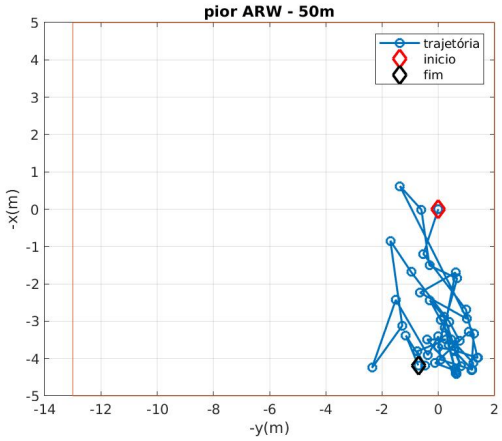


(c) Melhor trajetória para SP

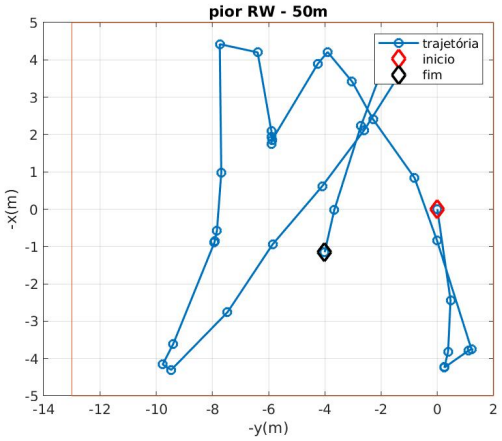
Figura 4.2: Melhores trajetórias para ambiente vazio e distância percorrida de 50m

Porém a Figura 4.1(a) apresenta um alto valor de desvio padrão para o algoritmo ARW, comparado com os outros algoritmos, o que leva a uma maior incerteza do desempenho obtido com o ARW. Por exemplo, a Figura 4.2 apresenta os melhores resultados dos 3 algoritmos para 50m, observe que o desempenho do ARW não é tão baixo, sobretudo ao compara-lo com o resultado do SP, presente na Figura 4.2(c).

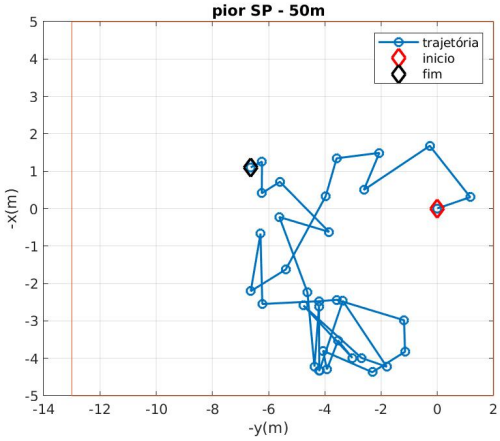
Além disso, a Figura 4.2 mostra que, mesmo com valores de média baixos, o ARW apresentou um melhor caso visualmente interessante, porém ao comparar as trajetórias com pior desempenho é claro como o ARW apresentou um maior número de ciclos. Para tal comparação, basta analisar a Figura 4.3, em que as Figuras 4.3(a)-4.3(c) apresentam os piores resultados para o ARW,RW e SP, respectivamente, para distância fixada de 50m.



(a) Pior trajetória para ARW



(b) Pior trajetória para RW

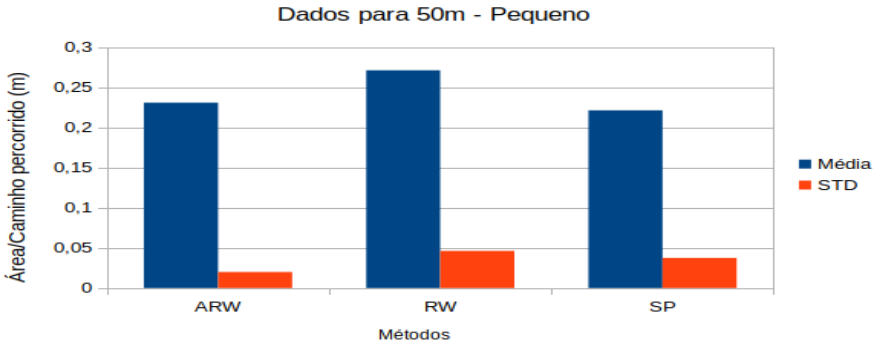


(c) Pior trajetória para SP

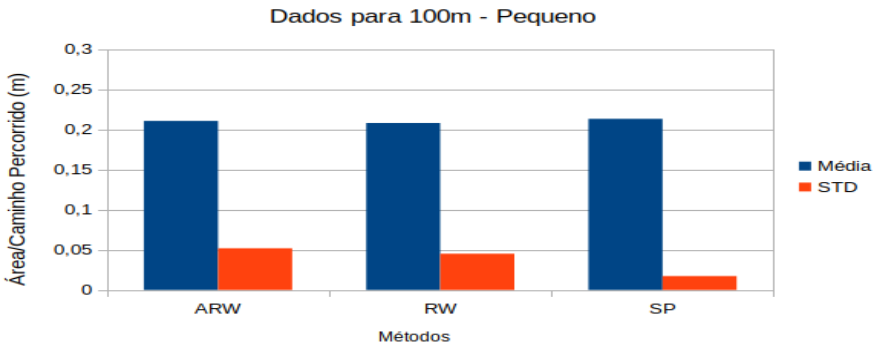
Figura 4.3: Piores trajetórias para ambiente vazio com distância percorrida de 50m

4.1.2 Ambiente Pequeno com Obstáculos

A seguir encontram-se os dados para o ambiente pequeno com obstáculos da Figura 3.6, em que as Figuras 4.4(a) e 4.4(b) apresentam dados de média e desvio padrão para 5 simulações quando fixado caminho de 50m e 100m, respectivamente. Os dados numéricos estão contidos na tabela 4.2 e finalmente, na Figura 4.5 estão as melhores ocorrências para o ambiente.



(a) Dados para distância fixa de 50m



(b) Dados para distância fixa de 100m

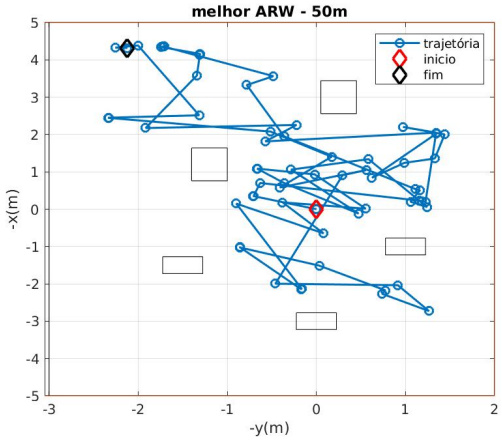
Figura 4.4: Dados para ambiente pequeno com obstáculos

A Figura 4.4 apresenta os dados de forma visual, uma vez que as escalas são as mesmas, porém a tabela 4.2 apresenta os dados numéricos.

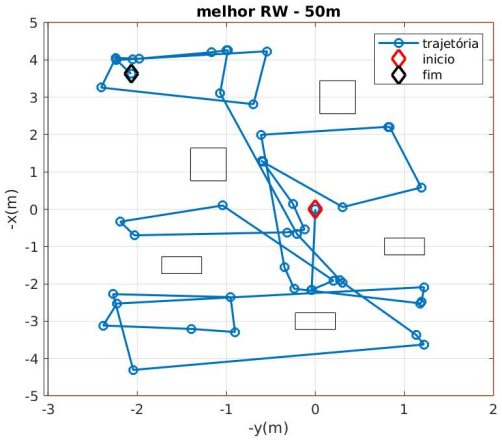
Método	Área/Distância para 50m	Área/Distância para 100m
ARW	$0.2309 \pm 0.0199m$	$0.2079 \pm 0.0450m$
RW	$0.2711 \pm 0.0461m$	$0.2079 \pm 0.0450m$
SP	$0.2212 \pm 0.0374m$	$0.2131 \pm 0.0172m$

Tabela 4.2: Valores médios para exploração do ambiente pequeno

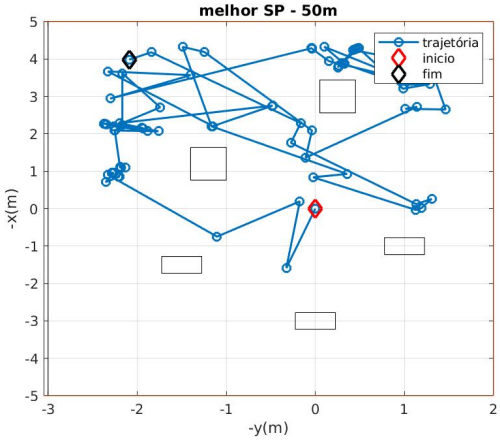
A Figura 4.5 apresenta os melhores resultados para o presente algoritmo, e aqui percebe-se um melhor desempenho visual do algoritmo RW, uma vez que além de possui menos repassagens, levou o robô a uma maior descoberta do ambiente como um todo, uma vez que passou por todos os cantos e interior.



(a) Melhor trajetória para ARW



(b) Melhor trajetória para RW



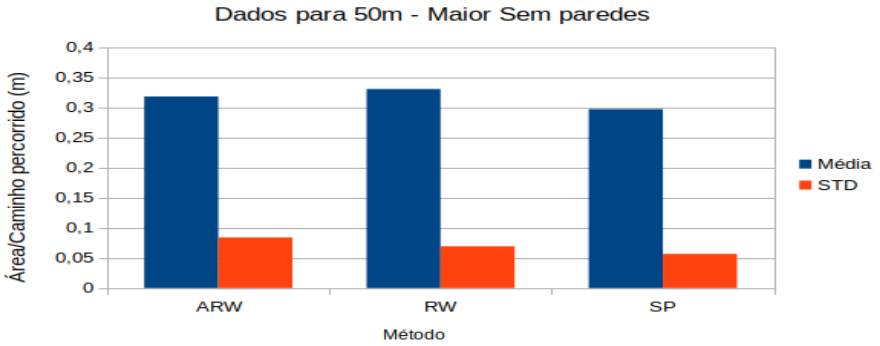
(c) Melhor trajetória para SP

Figura 4.5: Melhores trajetórias para ambiente pequeno e distância percorrida de 50m

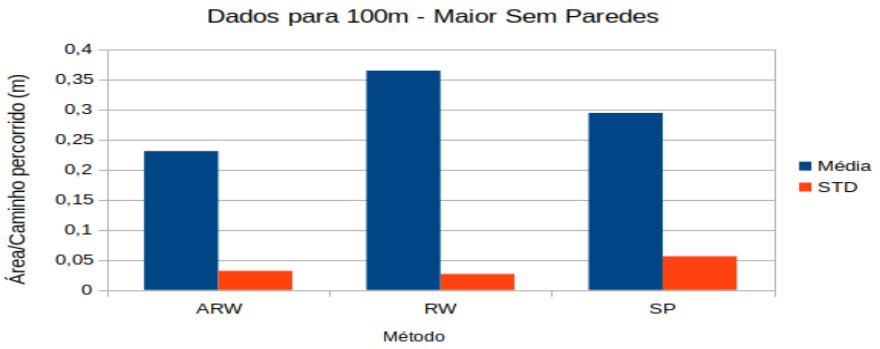
Ao comparar os dados do presente ambiente com o anterior observa-se que os valores de média da área coberta pela distância percorrida reduziram. Parte de tal ocorrido é justificável pelo tamanho do ambiente, uma vez que o presente ambiente é bem menor (1/3 da área), e dessa forma leva a um número maior de ciclos de trajetória e repassagens já em 50m, como mostra a Figura 4.2.

4.1.3 Ambiente Maior com Obstáculos

Assim como os anteriores, a seguir encontra-se os dados para as 5 simulações de cada algoritmo, para cada distância, em que a Figura 4.6 apresenta os valores em forma de gráfico, a tabela 4.3.



(a) Dados para distância fixa de 50m



(b) Dados para distância fixa de 100m

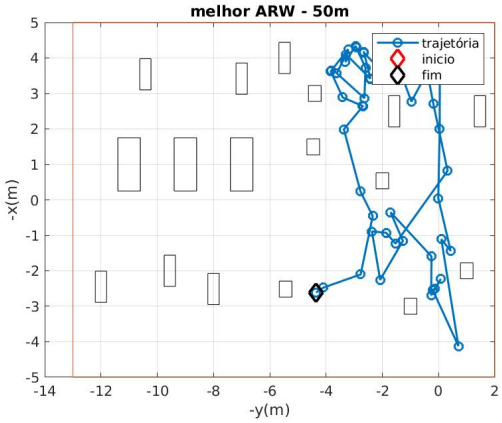
Figura 4.6: Dados para ambiente maior com obstáculos

No caso do ambiente maior e com obstáculos, para uma distância de 50m os resultados médios foram relativamente equilibrados como mostra a Figura 4.6(a), porém ambos apresentaram um nível mais elevado de incerteza se comparados com os outros métodos, o que indica que ocorreu muita variação entre as simulações, sugerindo que hora os algoritmos apresentaram bons resultados e hora tiveram um número elevado de ciclos em regiões locais do ambiente.

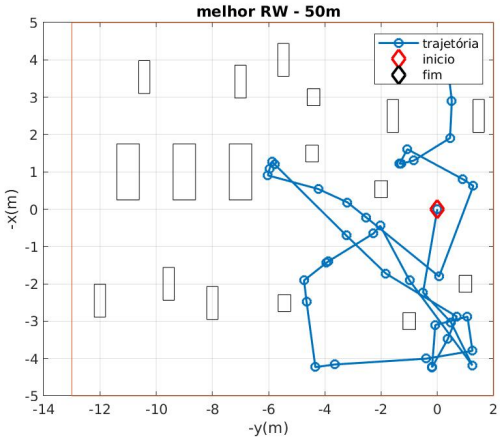
Já para a distância de 100m os resultados médios apresentaram uma menor desvio padrão, mesmo que o ARW apresentou um desempenho pior de média e o SP apresentou resultados extremamente próximos.

Método	Área/Distância para 50m	Área/Distância para 100m
ARW	0.3181 ± 0.0838	0.2306 ± 0.0317
RW	0.3303 ± 0.0692	0.3642 ± 0.0265
SP	0.2969 ± 0.0566	0.2940 ± 0.0560

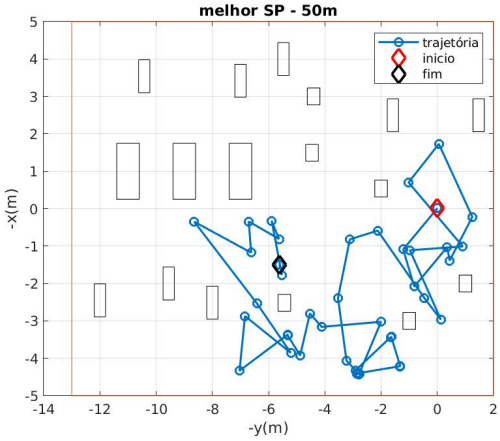
Tabela 4.3: Valores médios para exploração do ambiente maior com obstáculos e sem câmaras



(a) Melhor trajetória para ARW



(b) Melhor trajetória para RW

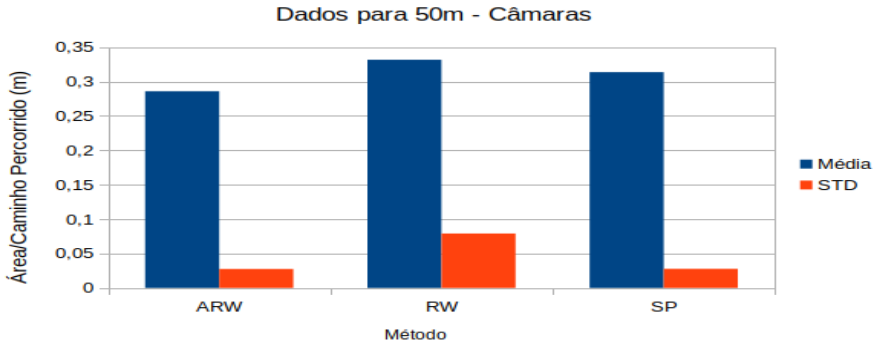


(c) Melhor trajetória para SP

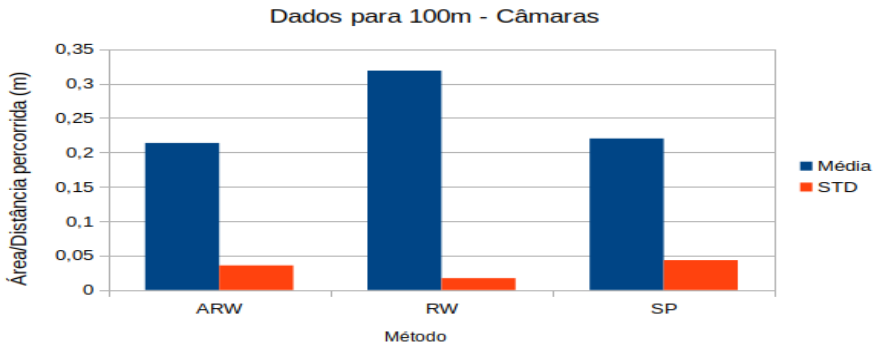
Figura 4.7: Melhores trajetórias para ambiente grande e sem câmaras, com distância percorrida de 50m

4.1.4 Ambientes Maior com Duas Câmaras

Finalmente, a Figura 4.8 apresenta os dados de média e desvio padrão para os algoritmos, e aqui, os resultados se aproximaram os obtidos nas análises anteriores, em que o RW apresentou melhores resultados de média. Porém, diferente dos outros, para distância de 100m o ARW ficou muito abaixo do RW.



(a) Dados para distância fixa de 50m

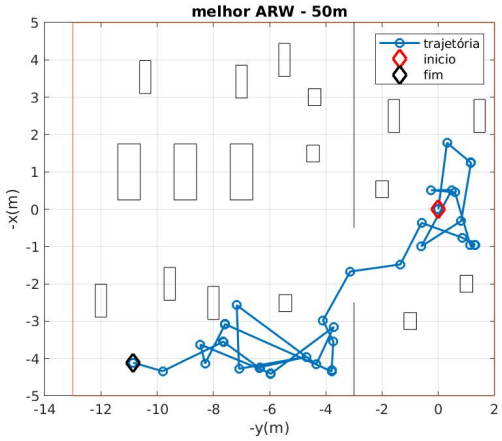


(b) Dados para distância fixa de 100m

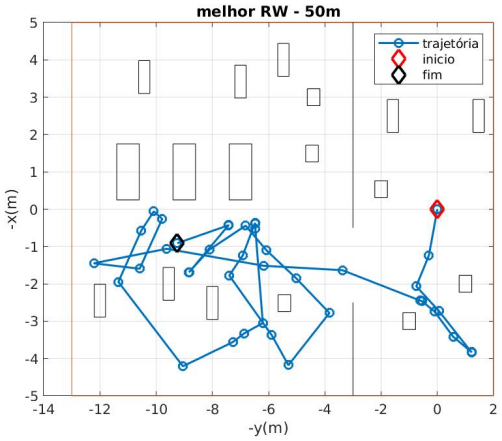
Figura 4.8: Dados para ambiente maior com obstáculos e duas câmaras

Método	Área/Distância para 50m	Área/Distância para 100m
ARW	0.2137 ± 0.0356	0.2857 ± 0.0276
RW	0.3187 ± 0.0171	0.3315 ± 0.0790
SP	0.2202 ± 0.0433	0.3135 ± 0.0277

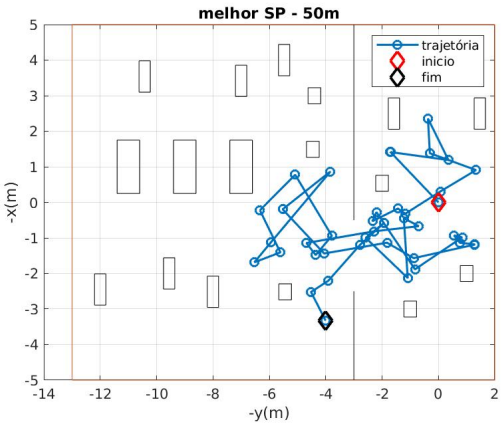
Tabela 4.4: Valores médios para exploração do ambiente maior e com câmaras



(a) Melhor trajetória para ARW



(b) Melhor trajetória para RW

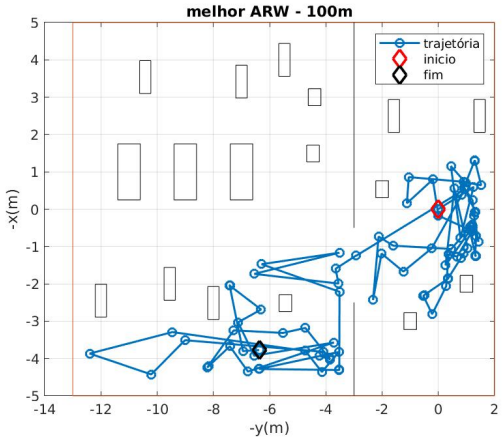


(c) Melhor trajetória para SP

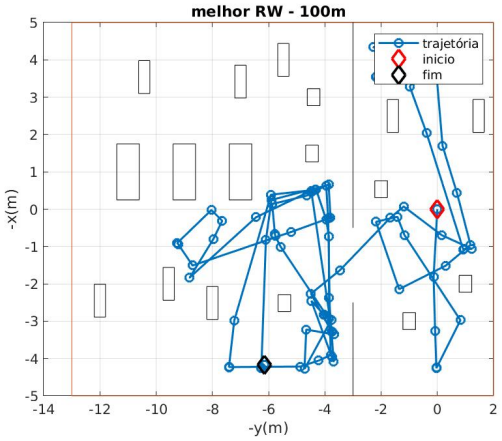
Figura 4.9: Melhores trajetórias para ambiente pequeno e distância percorrida de 50m

A Figura 4.9 apresenta os melhores resultado para trajetória máxima de 50m, tais resultados auxiliam de forma visual os resultados da Figura 4.8, que é possível ver como os algoritmos foram capazes de passar de uma câmara para outra ao menos uma vez, bem como que há poucos ciclos em tais simulações.

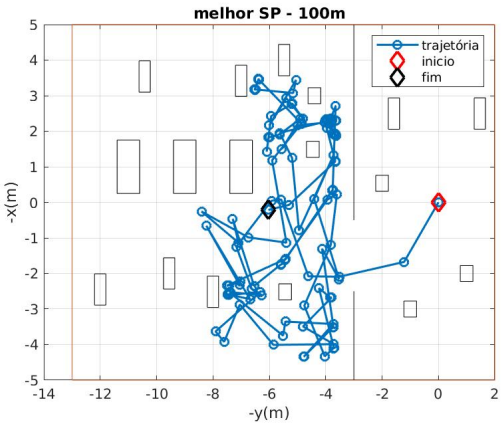
Da mesma forma, a Figura 4.10 apresenta os melhores resultados para o ambiente com câmaras, porém apresenta os melhores resultados para $100m$ de trajetória máxima.



(a) Melhor trajetória para SP



(b) Melhor trajetória para SP



(c) Melhor trajetória para SP

Figura 4.10: Melhores trajetórias para ambiente pequeno e distância percorrida de 100m

De forma geral, em ambos os ambientes o algoritmo RW apresentou melhores resultados que os demais como pode ser visto nos gráficos das Figuras 4.1. 4.4, 4.8 e 4.6. De fato, a análise visual mostrou que o algoritmo RW apresenta uma maior dispersão pelo mapa, visitando pontos mais distantes de forma mais rápida, o que é interessante para casos em que deseja-se ganho de nova informação, sobretudo informação de novos lugares. Porém o contrário que aconteceu com o ARW e o RW, de "demorar"em alguns lugares, apresentando pequenos incrementos e mudanças de direções é benéfico para casos de algoritmos em que é necessário a presença efetiva do robô em certas áreas do ambiente, como o caso de limpeza e agropecuária, em que é interessante para o robô, ao adentrar em uma região do ambiente que ele percorra localmente essa região.

Como esperado, por utilizarem de processos aleatórios, os algoritmos apresentaram ciclos e repassagens desnecessárias, levando a uma queda do número de desempenho médio dos algoritmos a partir da inserção dos obstáculos, uma vez que para o ambiente vazio os resultados médios se aproximavam de $0.5m$ presente na Figura 4.1, enquanto para os outros métodos os resultados se aproximaram de $0.35m$.

4.2 Robô em ambiente real

Além das simulações realizadas no ambiente Gazebo, o robô foi posto a prova em ambiente real, para verificar as dificuldades e comportamento perante ambiente mais complexo e dinâmico. Neste caso, uma vez que o robô não possui localização não foi possível fazer uma avaliação externa como a realizada nas simulações.

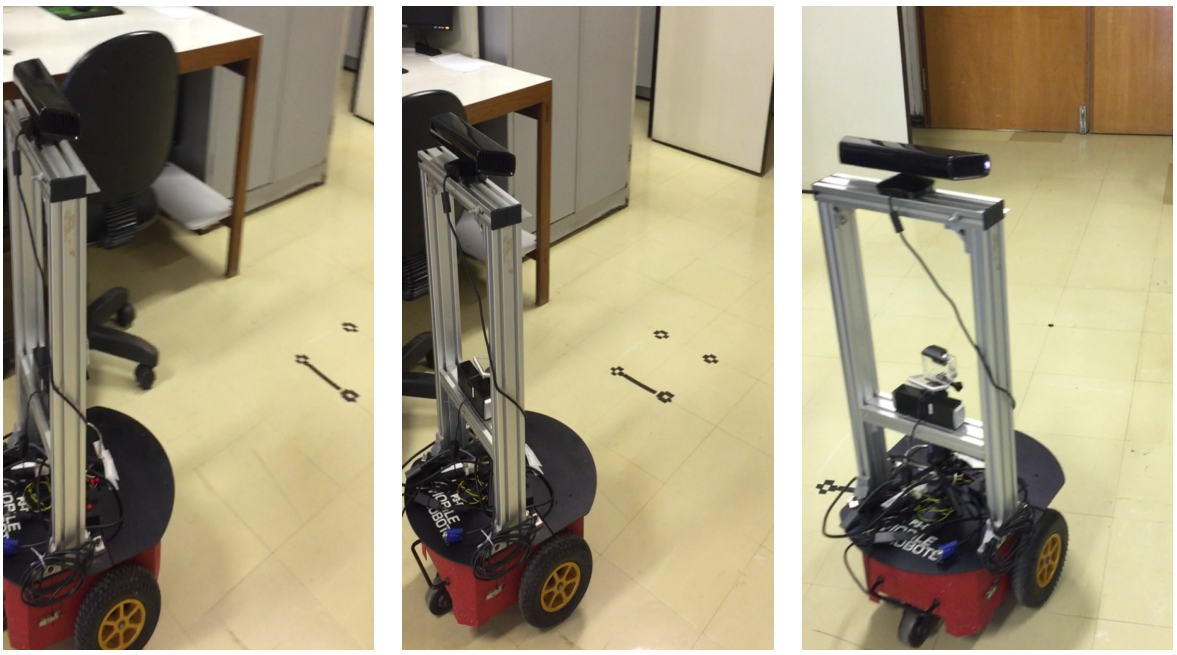
(a) Primeiro instante de exploração

(b) Segundo instante de exploração

(c) Terceiro instante de exploração

Figura 4.11: Teste de exploração em ambiente real e dinâmico

A Figura 4.11 apresenta uma sequência de imagens do robô explorando um ambiente real, o robô foi capaz de se movimentar sem colidir, mesmo quando pessoas passavam por seu caminho.



(a) Primeiro instante de exploração

(b) Segundo instante de exploração

(c) Terceiro instante de exploração

Figura 4.12: Teste de exploração em ambiente real e dinâmico em ambiente diferente

Novamente, a Figura 4.12 apresenta uma nova sequência de imagens do mesmo ambiente da Figura 4.11, porém em outra sala do mesmo ambiente. Em ambiente real o robô se movimentou como o esperado, e mesmo na presença de pessoas, que atravessaram em sua frente o robô não colidiu. Os vídeos da exploração em ambiente real estão disponíveis de forma pública [1].

[1] https://www.youtube.com/playlist?list=PLaeK4-sOfhRDVxB-h-ldQVxb7ZMQu2YJK

Capítulo 5

Conclusões

5.1 Resumo

Atividades como exploração e cobertura estão em muitos casos associadas a representações do ambiente, e tais atividades podem ser impraticáveis quando necessitam de representação prévia, ou na presença de ambientes dinâmicos ou em ambientes de maior escala, o que pode prejudicar o desempenho da representação ambiental bem como o da exploração. Aqui foram feitas avalizações de algoritmos baseados diferentes movimentações aleatórias, uma estrutura local foi utilizada para evitar repassagens. O robô movimenta-se sem nenhuma representação gráfica, utilizando informações de distância da nuvem de pontos e dos sonares para evitar obstáculos.

Dentre os algoritmos comparados, o passeio aleatório RW que utiliza de sorteio de ângulos obteve melhores resultados na grande maioria dos casos, em que a avaliação utilizada foi a razão da área coberta pelo robô pela distância por ele percorrida. Porém, por se tratar de algoritmos de incrementos aleatórios, os resultados apresentaram resultados com ocorrência de ciclos, sobretudo em regiões locais com muito obstáculos, mesmo utilizando uma estrutura para evitar repassagem. Tais constatações são refletidas em um nível relativamente alto de desvio padrão em algumas simulações.

Em ambiente real o robô é capaz de se movimentar e explorar sem ocorrência de colisões, mesmo na presença de dinâmica ambientais tais como interações com pessoas passando pelo robô ou mudanças estruturais no ambiente, porém nenhuma avaliação foi feita em ambiente real, de forma que não pode-se afirmar sobre sua eficiência em ambiente real.

Os códigos desenvolvidos nesse trabalho estarão disponíveis em forma pública na plataforma *GitHub* pessoal[1] e do LARA[2].

[1] `https://github.com/Abrantex/tg_matheus`
[2] `https://github.com/lara-unb`

5.2 Perspectivas Futuras

Uma vez obtidos os resultados aqui relatados, espera-se em um futuro ser capaz de avaliar de forma assertiva os resultados obtidos em ambiente real, utilizando algum sistema de localização ou medindo a posição do robô com alguma ferramenta externa tal como uma câmera calibrada. Além disso, espera-se comparar os modelos obtidos com algoritmos de outra complexidade tais quais àqueles que utilizam de padrão de movimentação em *zig-zag*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] QIAN, K.; SONG, A. Autonomous navigation for mobile robot based on a sonar ring and its implementation. In: IEEE. *Instrumentation and Control Technology (ISICT), 2012 8th IEEE International Symposium on*. London, UK, 2012. p. 47–50.

[2] LAVALLE, S. M. *Planning algorithms*. Cambridge, USA: Cambridge university press, 2006.

[3] ADORNO, B. V. *Planejamento probabilístico de rotas no espaço de configuração e sua aplicação em robótica móvel*. Dissertação (Mestrado) — UnB, Brasília, 2008.

[4] HAN, J. et al. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, IEEE, v. 43, n. 5, p. 1318–1334, 2013.

[5] STONE, E. E.; SKUBIC, M. Fall detection in homes of older adults using the microsoft kinect. *IEEE J. Biomedical and Health Informatics*, v. 19, n. 1, p. 290–301, 2015.

[6] LANGE, B. et al. Development and evaluation of low cost game-based balance rehabilitation tool using the microsoft kinect sensor. In: IEEE. *Engineering in medicine and biology society, EMBC, 2011 annual international conference of the IEEE*. Boston, MA, USA, 2011. p. 1831–1834.

[7] KAMEYAMA, N.; HIDAKA, K. A sensor-based exploration algorithm for autonomous map generation on mobile robot using kinect. In: IEEE. *Control Conference (ASCC), 2017 11th Asian*. Gold Coast, QLD, Australia, 2017. p. 459–464.

[8] QUIGLEY, M. et al. Ros: an open-source robot operating system. In: *ICRA workshop on open source software*. Kobe, Japan: IEEE, 2009. v. 3, n. 3.2, p. 5.

[9] YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: IEEE. *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. Monterey, USA, 1997. p. 146–151.

[10] YAMAUCHI, B. Frontier-based exploration using multiple robots. In: ACM. *Proceedings of the second international conference on Autonomous agents*. Minneapolis, Minnesota, USA, 1998. p. 47–53.

[11] THRUN, S.; BURGARD, W.; FOX, D. Probabilistic robotics. In: *Intelligent robotics and autonomous agents*. Cambridge, USA: MIT press, 2005.

[12] HOLZ, D. et al. Evaluating the efficiency of frontier-based exploration strategies. In: VDE. *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*. Munich, Germany, 2010. p. 1–8.

[13] TAYLOR, C. J.; KRIEGMAN, D. J. Vision-based motion planning and exploration algorithms for mobile robots. *IEEE Transactions on robotics and Automation*, IEEE, v. 14, n. 3, p. 417–426, 1998.

[14] BATALIN, M. A.; SUKHATME, G. S. Efficient exploration without localization. In: IEEE. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Taipei, Taiwan, Taiwan, 2003. v. 2, p. 2714–2719.

[15] TOVAR, B.; MURRIETA-CID, R.; LAVALLE, S. M. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, IEEE, v. 23, n. 3, p. 506–518, 2007.

[16] LAGUNA, G. et al. Exploration of an unknown environment with a differential drive disc robot. In: IEEE. *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. Hong Kong, China, 2014. p. 2527–2533.

[17] KATSEV, M. et al. Mapping and pursuit-evasion strategies for a simple wall-following robot. *IEEE Transactions on Robotics*, IEEE, v. 27, n. 1, p. 113–128, 2011.

[18] ROMERO, R. A. et al. Robótica móvel. *São Paulo: LTC*, 2014.

[19] BALCH, T.; ARKIN, R. Avoiding the past: A simple but effective strategy for reactive navigation. In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. Atlanta, GA, USA: IEEE, 1993. p. 678–685.

[20] CHOSET, H.; PIGNON, P. Coverage path planning: The boustrophedon cellular decomposition. In: SPRINGER. *Field and service robotics*. London, England, 1998. p. 203–209.

[21] ZHANG, H.; WANG, W. et al. A topological area coverage algorithm for indoor vacuuming robot. In: IEEE. *Automation and Logistics, 2007 IEEE International Conference on*. Jinan, China, 2007. p. 2645–2649.

[22] JIA, D. et al. Coverage path planning for legged robots in unknown environments. In: IEEE. *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. Lausanne, Switzerland, 2016. p. 68–73.

[23] LATOMBE, J.-C. *Robot motion planning*. New York, USA: Springer Science & Business Media, 2012.

[24] RENZAGLIA, A.; MARTINELLI, A. Potential field based approach for coordinate exploration with a multi-robot team. In: IEEE. *Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on*. Bremen, Germany, 2010. p. 1–6.

[25] MURPHY, R.; MURPHY, R. R. *Introduction to AI robotics*. Cambridge, USA: MIT press, 2000.

[26] PEARSON, K. The problem of the random walk. *Nature*, Nature Publishing Group, v. 72, n. 1867, p. 342, 1905.

[27] KREJSA, J.; VECHET, S. Covering the working space of mobile robot. In: IEEE. *Mechatronics-Mechatronika (ME), 2014 16th International Conference on*. Brno, Czech Republic, 2014. p. 469–472.

[28] MORLOK, R.; GINI, M. Dispersing robots in an unknown environment. In: *Distributed Autonomous Robotic Systems 6*. Tokyo, Japan: Springer, 2007. p. 253–262.

[29] CARPIN, S.; PILLONETTO, G. Motion planning using adaptive random walks. *IEEE Transactions on Robotics*, IEEE, v. 21, n. 1, p. 129–136, 2005.

[30] STOWERS, J.; HAYES, M.; BAINBRIDGE-SMITH, A. Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In: IEEE. *Mechatronics (ICM), 2011 IEEE International Conference on*. Istanbul, Turkey, 2011. p. 358–362.

[31] KHOSHELHAM, K.; ELBERINK, S. O. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, Molecular Diversity Preservation International, v. 12, n. 2, p. 1437–1454, 2012.

[32] SIEGWART, R. et al. *Introduction to autonomous mobile robots*. Cambridge, Massachusetts: MIT press, 2011.

[33] SHAMAH, B. et al. Steering and control of a passively articulated robot. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Sensor Fusion and Decentralized Control in Robotic Systems IV*. Boston, MA, United States, 2001. v. 4571, p. 96–108.

ANEXOS

I. PROGRAMAS UTILIZADOS

Os códigos desenvolvidos no presente trabalho estão disponíveis em repositório aberto que contém arquivo *README* que traz informações para execução dos códigos[1]. Os códigos foram desenvolvidos segundo ambiente de trabalho ROS, versão *kinetic* em sistema operacional Ubuntu 16.04.

[1] `https://github.com/Abrantex/tg_matheus`